

METHODOLOGY ARTICLE

Open Access



String kernels for protein sequence comparisons: improved fold recognition

Saghi Nojoomi¹ and Patrice Koehl^{2*}

Abstract

Background: The amino acid sequence of a protein is the blueprint from which its structure and ultimately function can be derived. Therefore, sequence comparison methods remain essential for the determination of similarity between proteins. Traditional approaches for comparing two protein sequences begin with strings of letters (amino acids) that represent the sequences, before generating textual alignments between these strings and providing scores for each alignment. When the similitude between the two protein sequences to be compared is low however, the quality of the corresponding sequence alignment is usually poor, leading to poor performance for the recognition of similarity.

Results: In this study, we develop an alignment free alternative to these methods that is based on the concept of string kernels. Starting from recently proposed kernels on the discrete space of protein sequences (Shen et al, *Found. Comput. Math.*, 2013,14:951-984), we introduce our own version, SeqKernel. Its implementation depends on two parameters, a coefficient that tunes the substitution matrix and the maximum length of *k-mers* that it includes. We provide an exhaustive analysis of the impacts of these two parameters on the performance of SeqKernel for fold recognition. We show that with the right choice of parameters, use of the SeqKernel similarity measure improves fold recognition compared to the use of traditional alignment-based methods. We illustrate the application of SeqKernel to inferring phylogeny on RNA polymerases and show that it performs as well as methods based on multiple sequence alignments.

Conclusion: We have presented and characterized a new alignment free method based on a mathematical kernel for scoring the similarity of protein sequences. We discuss possible improvements of this method, as well as an extension of its applications to other modeling methods that rely on sequence comparison.

Keywords: Protein sequence, Kernel, Alignment free methods

Background

The overall principles behind the translation of a gene sequence into an active protein structure are theoretically well understood. The sequence of nucleotides that forms a gene is first translated into a sequence of amino acids, based on the genetic code. The corresponding linear chain of amino acids becomes active only after it folds into a three-dimensional shape, the so-called tertiary structure of the protein (though there is mounting evidence that there is a whole class of “unstructured” proteins that are biologically relevant [1]). However, while we can easily observe experimentally how the linear chain folds, we do

not understand the principles underlying that transition. For instance, the rules that reveals the three-dimensional structure of a protein from its amino acid sequence have not yet been unraveled. Finding these rules is in fact one of the “holy grails” in molecular biology, namely the protein structure prediction problem [2, 3]. Efforts to solve this problem currently emphasize protein sequence analysis, as a consequence of the wealth of sequence data that is available. We know the primary sequences of many proteins, with currently more than 65,000,000 sequences in the TrEMBL database [4], we know in some cases the corresponding three-dimensional structures of these proteins, with more than 110,000 in the Protein Data Bank [5], and we have information on their biological activities, with more 550,000 sequences that have been annotated in SwissProt [6] (all numbers correspond to August 2016). However, we are still far from understanding all

*Correspondence: pakoehl@ucdavis.edu

²Department of Computer Science and Genome Center, 1, Shields Avenue, 95616 Davis, CA, USA

Full list of author information is available at the end of the article

the information encoded in these databases. In this paper, we address the problem of protein sequence comparison in the context of protein fold recognition, and show that a new string kernel drastically improves the latter compared to traditional methods based on sequence alignment.

Amino acids are usually described using a one-letter code, and protein sequences are correspondingly represented as strings of letters. This representation has proved very useful, especially in the context of sequence alignment [7, 8] that is usually performed using string-matching algorithms [9]. Those methods represent the overwhelming majority of methods used for sequence comparisons in bioinformatics. When comparing two sequences, they proceed in two steps, first the generation of the alignment between the two sequences, then the derivation of a statistical score for that alignment. They rely on a weighting scheme to measure the cost of matching pairs of amino acids. Many such weights have been proposed, from substitution matrices derived from evolution models such as the PAM matrices [10] and the BLOSUM matrices [11], to matrices that capture physico-chemical properties of amino acids [12]. Using this score, an alignment is derived following a dynamic programming algorithm, either the local method of Smith and Waterman [13], or the global method of Needleman and Wunsch [14]. This alignment is then scored by summing the individual weights of the matching pairs of amino acids and adding penalties for the presence of gaps. It should be noted that this score is not a metric in sequence space. Statistical methods have been developed to assess the significance of such scores, both for gapped and non-gapped alignments (see for example [15]). Such statistical scores are widely used for the identification of homologous sequences or for fold recognition. It has been shown however that those scores are efficient for both tasks for sequences with high similarities, but often fail for dissimilar sequences (see for example [16]). Extensions to pair-wise alignment methods have been proposed to alleviate this problem, such as those based on multiple sequence alignments and profiles [17], and those based on Hidden Markov Models [8]. While those show improved sensitivity, they remain prone to the problems related to the construction and use of alignments.

To circumvent the shortcomings of the alignment-based methods described above, many “alignment-free” methods have been proposed over the past three decades (for review, see [18–20]). Most of these methods compute the frequencies of words of a fixed length, k , also denoted as k -mers or k -grams, depending on the authors. Once the frequency distribution functions of such k -mers have been computed for two sequences, the distance between those two sequences is assimilated to the distance between those distributions, using different definitions of distance

[19, 21]. Other methods identify word matches of different lengths [22, 23]. One such method, the *average common substring approach*, identifies for each position i in one sequence the longest substring starting at i that is also present in the other sequence. The average lengths of those substrings over all positions i is a measure of similarity of the two sequences that can be converted into a distance [22]. All those methods are based on exact word matches. Exact matches however are bound to have limitations, due to strong correlations between amino acids at neighboring positions. A solution to the limitations of exact matches is the *spaced seeds* method that defines patterns with “*match*” and possible “*don't care*” positions, using the vocabulary introduced by the authors [24–27].

Another class of alignment-free methods for comparing protein sequences that are directly relevant to this work includes string kernel based methods, originally defined in the context of support vector machines (SVM) [28]. SVM are machine learning algorithms that are designed to learn a rule for discrimination from a set of samples with two (or more) labels. This rule can then be applied to predict the label of any new sample. A key element of any SVM implementation is the kernel function that is used to quantify the similarity between any pair of samples. The simplest kernel function is the dot product between vectors of features that represent the samples. The first applications of SVM with a kernel function that is used to compute the similarity between two protein sequences were based on an extension of the dot product concept. Jakkola et al [29] used a generative Hidden Markov Model (HMM) on a set of proteins to generate a vector representation of each protein sequence (the so-called Fisher score vector). The kernel is then defined as a dot product between the corresponding Fisher vectors. Lodhi and colleagues introduced a string kernel that counts the number of occurrences of subsequences of a fixed length in the two strings that are compared [28]. The SVM-pairwise method [30] consists of describing each sequence with a vector of pairwise similarity scores for all domains in the training set (where the similarity score is the E-value of the Smith-Waterman pairwise sequence alignment), and defines the kernel to be the dot product between these vector representations. The spectrum kernel [31] and the mismatch kernel [32] measure the similarity between protein sequences by quantifying the number of similar short substrings (i.e. k -mers of fixed lengths, typically between 4 and 6 amino acids) they share. These two kernels bear similarity with the word-based alignment-free methods described above. The weighted degree kernels extend those kernels by considering weighted sums of the individual kernels obtained with fixed length k -mers [33, 34]. The local alignment kernel of Saigo et al [35] was designed to mimic the score generated by a Smith and Waterman pairwise alignment

method, with the proper mathematical foundations to guarantee that it is a true kernel. More recently, Smale and co-workers expanded the local alignment kernel by considering all possible alignments of k -mers between the two sequences of interest, for all possible k values, ignoring gaps when aligning the k -mers [36]. All the kernels listed here (as well as others that we have most likely missed), have been tested in classification problems as part of a machine learning algorithm (usually SVM), with various levels of success.

This paper draws from the concept of string kernels listed above. It describes an alignment-free method for protein sequence comparison that is based on the string kernel introduced by Smale and collaborators [36]. In contrast with the previous studies on string kernels, we do not include at this stage our string kernel into any learning algorithms. Instead, we assess directly its ability to classify proteins into structural folds based on sequence information only. We note that the string kernel we consider (which we refer to as SeqKernel) depends on two parameters, in addition to the substitution matrix it uses to score matches of pairs of amino acids (see below). We provide an exhaustive analysis of the effects of these two parameters on the performance of the kernel for fold recognition. Such an analysis, which is necessary as a first step to improving string kernel methods, was only partially included in the presentations of the equivalent kernels defined by Saigo et al [35], and by Smale and co-workers [36]. In the latter study for example, one parameter, β (see below for details), was fixed to 0.11, without further explanation, while the second parameter, k_{max} , the maximum length of the k -mers considered, is set to a small number for computational considerations. Finding how those parameters influence the performance of the kernel is the main focus of this paper. To perform this analysis, we consider different datasets of proteins that belong to different structural folds, as defined by CATH [37]. These proteins were selected such that their sequences share little sequence similarities. We classified these proteins using the similarity measure provided by the string kernels. The classification is then compared with the corresponding results obtained using the scores of pairwise sequence alignments and the scores of structural alignments of those proteins. Ultimately, we have observed that sequence alone provides poor separation of the different folds. We show in contrast that with the right choice of parameters, use of the string kernel similarity measure significantly improves classification accuracy. In addition, we illustrate the use of SeqKernel to reconstruct a phylogenetic tree of RNA polymerases, in line with the current development of alignment-free methods for phylogenomics that are designed to remove problems inherent to (multiple) sequence alignments (see for example [38–40]).

The paper is organized as follows. In “Methods” section, we describe the string kernel, while in “Results and Discussion” section we describe the databases we have generated to assess the performance of our string kernel for fold recognition. In the Results section, we provide a comprehensive analysis of the parameters that defined the kernel and finally we conclude the paper with a discussion on future developments of the string kernel approach.

Methods

A string kernel for alignment-free sequence comparison

The string kernel considered here is inspired by the convolution string kernels introduced by D. Haussler [41], adapted by Saigo et al [35] as the local alignment kernels, and later by Smale and co-workers [36]. We provide here the key elements of its construction. Readers should refer to the original papers for a more detailed presentation, notably for the proofs of the mathematical properties that are relevant to kernels in general.

Notations Let X be a finite set of size n . A kernel K is a symmetric function from $X \times X$ to \mathbb{R} such that the Gram matrix G of size $n \times n$ defined by $G(i, j) = K(x_i, x_j)$ is symmetric, positive, and definite. Let \mathcal{A} be the set of the standard twenty amino acids found in proteins. A protein sequence S is a string of elements from \mathcal{A} . We note $|S|$ the length of S .

A kernel for amino acid pairs. Let SA be a function from $\mathcal{A} \times \mathcal{A}$ to \mathbb{R} , such that $SA(i, j)$ measures the similarity of the amino acids i and j , and let SM be the Gram matrix associated to SA . Examples of SM include the matrices representing the raw data of any BLOSUM matrices [42], namely the raw counts of how often amino acid i is substituted by amino acid j in a set of selected protein sequence alignments that is then normalized by considering its row sums $P(i)$:

$$P(i) = \sum_{j=1}^{20} SM(i, j)$$

$$SM2(i, j) = \frac{SM(i, j)}{P(i)P(j)} \quad (1)$$

We have checked that when SM is a raw count BLOSUM matrix, then $SM2$ is symmetric, positive, and definite. Note that when SM is such a raw count matrix, the corresponding BLOSUM matrix BL is defined as $BL(i, j) = \text{round}(\log_2(SM2(i, j)))$, where round is the function that rounds a real number to its nearest integer. Given a strictly positive real number β , we define the function $K_1 : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}$ as:

$$K_1(i, j) = SM2(i, j)^\beta \quad (2)$$

K_1 is a kernel function on $\mathcal{A} \times \mathcal{A}$, as long as $SM2$ is symmetric, positive, definite, and β is strictly positive. The same definition was used in [36].

A kernel for comparing two strings of the same length.

Let k be a strictly positive integer and let \mathcal{A}^k be the k -th Cartesian power of \mathcal{A} . An element of \mathcal{A}^k is a string of k letters taken from \mathcal{A} , it is usually referred to as a k -mer, or a k -gram. Let $u^k = (u_1, \dots, u_k)$ and $v^k = (v_1, \dots, v_k)$ be two k -mers in \mathcal{A}^k . The function K_2^k defined by:

$$K_2^k(u^k, v^k) = \prod_{i=1}^k K_1(u_i, v_i) \tag{3}$$

is a kernel on \mathcal{A}^k , the set of all k -mers. We note that K_2^k is a convolution kernel [41].

A kernel for computing protein sequence similarity.

Let $S = (s_1, \dots, s_n)$ and $T = (t_1, \dots, t_m)$ be two protein sequences; both are strings, with $S \in \mathcal{A}^n$ and $T \in \mathcal{A}^m$. Let u^k and v^k be substrings of length k (i.e. k -mers) of S and T respectively. u^k and v^k are considered contiguous, i.e. we do not allow gaps. There are therefore $n - k + 1$ and $m - k + 1$ such substrings in S and T , respectively. We define

$$K_3^k(S, T) = \sum_{u^k \in S} \sum_{v^k \in T} K_2^k(u^k, v^k) \tag{4}$$

and

$$K_3(S, T) = \sum_{k=1}^p K_3^k(S, T). \tag{5}$$

where the summation extends to $p = \min(n, m)$, with n and m the lengths of the two sequences that are compared. Finally, we define the correlation kernel \hat{K}_3 as:

$$\hat{K}_3(S, T) = \frac{K_3(S, T)}{\sqrt{K_3(S, S)K_3(T, T)}} \tag{6}$$

Following [41] and [36], we make the following remarks:

- i) The input kernel matrix SM is not a traditional substitution matrix, as it does not involve applying the logarithm function on the probability measures. While the latter is needed to make scores additive, a necessary condition for the use of dynamic programming algorithms in generating pairwise sequence alignment, it is not needed for the string kernel we use here. Note that this differs from the local alignment kernel, which is designed to mimic pairwise alignment.
- ii) The kernel K_3 is computed as an unweighted sum of the individual kernels K_3^k that are computed with a fixed k value. We could have considered a weighted sum instead, similar in spirit to the weighted degree kernels [34]. We leave this option for future work.

- iii) The summation in Eq. 5 extends to the length of the smallest sequence. This summation can be truncated however to only include k -mers up to a fixed length set to be k_{max} . This will be discussed in details.
- iv) \hat{K}_3 is a kernel as long as $SM2$ and β (which define K_1) are definite positive, and strictly positive, respectively. Note that for any kernel K_1 defined on $\mathcal{A} \times \mathcal{A}$, we can define a correlation string kernel \hat{K}_3 .
- v) As defined, \hat{K}_3 does not consider gap penalties, or even gaps. We consider this as an advantage, as it does reduce the number of parameters that defines \hat{K}_3 .
- vi) The string kernel \hat{K}_3 is a similarity measure in the space of sequences. Notice that for all sequences S , $\hat{K}_3(S) = 1$. This similarity measure can be transformed into a distance measure, using

$$D(S, T) = \sqrt{\hat{K}_3(S, S) + \hat{K}_3(T, T) - 2\hat{K}_3(S, T)} = \sqrt{2 - 2\hat{K}_3(S, T)}. D(S, T) \text{ takes values between } 0 \text{ and } \sqrt{2}.$$

Implementing the string kernel

Equation 5 above leads to a simple, naive algorithm for computing the K_3 kernel for two sequences S and T : for any length k , generate all $n - k + 1$ and $m - k + 1$ k -mers in S and T , and perform the $(n - k + 1)(m - k + 1)$ corresponding K_2^k kernel evaluations using Eq. 3. Such an implementation however would come at a high computing cost, namely $O(nmk_{max}^2)$, where n and m are the lengths of S and T and k_{max} is the longest k -mer considered. We notice however that for any k -mers u^k and v^k , we have:

$$K_2^k(u^k, v^k) = K_2^{k-1}(u^{k-1}, v^{k-1})K_1(u^k, v^k) \tag{7}$$

where u^{k-1} is the string of length $k - 1$ formed from the $k - 1$ first components of u^k (with a similar definition for v^{k-1}). This simple recursion formula leads to a more efficient algorithm for computing the string kernel K_3 of order $O(nmk_{max})$. We note that this complexity remains large for protein sequence comparison.

Datasets of protein sequences

Our first dataset includes 10,619 domains from the CATH [37] v4.0 domains, each with a CATH classification. As we focus on protein structure fold prediction, we consider the first three layers of the CATH classification, namely, Class (C), Architecture (A), and Topology (T). A set of structures with the same C, A, and T defines a fold. By using a set of structures with significant sequence diversity, we ensure that the data is duplicate-free. Such a filter when selecting the sequences also assures that the problem of detecting structural similarity is non-trivial. The 10619 structures were selected as follows: (i) Randomize

the list of 235,858 CATH v4.0 domains; (ii) Start with the first domain on the randomized list, and remove from the list all domains that share significant sequence similarity with it (FASTA [43] SSEARCH E-value $< 10^{-4}$, with the alignments computed with the BLOSUM62 substitution matrix [42], and gap penalties of -11 (opening) and -1 (extension)). (iii) Repeat step (ii) with all domains in the list that have not yet been removed, until there are no domains left for selection. The set of 10619 domains resulting from this procedure is referred to as CATH40e4.

There are 1363 folds in CATH40e4, many of which only include a single representative (734). To improve statistical analysis and remove possible biases from those folds with a low number of representative, we selected five of the most populated folds in CATH40e4 as a more specific test set, including at least one fold from each CATH class: CATH fold 1.10.10, a fully α fold (arc repressor, 381 members), CATH fold 2.60.40, a fully β fold (immunoglobulin-like, 555 members), and three alternating α/β folds: 3.20.20, (TIM-like, 251 members), 3.30.70, (two layer sandwich, 368 members) and 3.40.50 (Rossmann fold, 1278 members). The same five folds were previously used in other studies [44, 45], but with different members as based on a different version of CATH. Overall, these five folds consist of a total of 2833 proteins (set CATH2833). Table 1 provides general information about the distributions of proteins in those five folds, while Fig. 1 illustrates the geometries of those folds, using the structure representatives defined by CATH.

As seen in Table 1, the proteins in the five folds included in CATH2833 vary significantly in length. To assess the importance of length differences when computing the string kernel between two sequences, we generated a subset of CATH2833, CATH793, that contains all proteins whose lengths are between 120 and 180 amino acids. General information about those proteins is available in Table 1.

Finally, we extracted a second dataset from CATH40e4 that contains all folds with at least 40 representatives, excluding the five folds included in CATH2833. There are

40 of those folds, covering all three classes of proteins, and 13 architectures. This dataset contains a total of 3744 proteins and is referred to as CATH3744.

ROC analysis of protein fold recognition

We quantify the effectiveness of a distance measure in identifying correctly if two sequences correspond to proteins that belong to the same CATH group using the receiver operating characteristic (ROC) analysis [46]. In the following, a “group” may be the class of the sequence (i.e. α , β , or α/β), the architecture of the corresponding protein structure, or the topology of the protein structure, as defined by CATH. A pair of protein sequences is defined as similar, or “positive”, if the sequences are members of the same group, and “negative” otherwise. All pairs of protein sequences in a dataset are then compared using a similarity measure. For varying thresholds of the measure considered, all pairs that fall below the threshold are assumed to be positive, and all above it are considered negative. The pairs that agree with the assumed standard are then called true positives (TP), while those that do not are deemed false positives (FP). A ROC analysis is set to measure the rate of TP as a function of the rate of FP. The “quality” of the similarity measure, namely its ability to separate positive pairs from negative pairs, is then scored using the Area Under the corresponding Curve, namely the AUC. An AUC score of 1 would indicate that all TP are detected first: this is consistent with an ideal measure. An AUC score of 0.5 indicates that TP and FP follow the first diagonal: they therefore appear at the same rate, indicating that the measure is not discerning.

When the number of groups considered is larger than two, we have performed two types of ROC analyses. In the first approach, we label a pair of proteins that belong to a given group as positive, independent of the group considered. In this case, an average behavior over all possible groups is derived. In the second approach, only pairs of proteins that belong to a specific group are called positive. This provides a group specific ROC analysis.

Fold classification experiments

The ROC analysis described above ranks the distance measures of pairs of protein sequences and checks if this ranking is consistent with an existing classification; it is not designed to achieve the classification itself. We have expanded the ROC analysis to the actual issue of fold recognition by carrying out a second set of experiments. Each experiment involves a data set of protein sequences, D , a level of classification with groups, G , and a distance measure, d . We start by dividing randomly the sets of sequences in D into two subsets of approximately equal size. The first subset defines a training set, while the second subset corresponds to the test set. A test sequence is assigned a class by mapping it to the group of its

Table 1 Statistics of the sequences included in the two datasets

CATH fold ID	Dataset			
	CATH2833		CATH793	
	N ^a	L (SD) ^b	N	L (SD)
1.10.10	381	79 (26)	36	135 (10)
2.60.40	555	110 (29)	130	140 (16)
3.20.20	251	294 (69)	2	157 (14)
3.30.70	368	182 (59)	52	141 (18)
3.40.50	1278	153 (77)	573	151 (17)

^aNumber of proteins in the fold

^bMean (standard deviation) of the lengths of the proteins in the fold

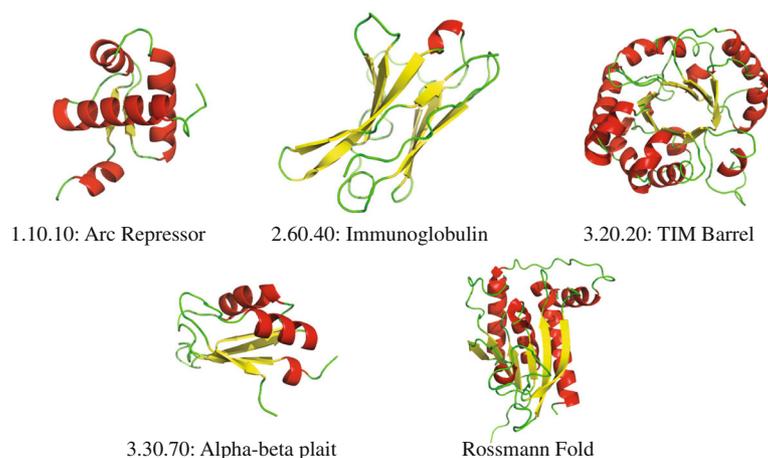


Fig. 1 Representatives of the five fold classes in our test set CATH2833. The arc repressor mutant, subunit A fold (CATH 1.10.10) is a common orthogonal helix bundle, found, for example, in the Zb domain from the RNA editing human enzyme ADAR1 (CATH code 1xmka00). The immunoglobulin-like fold is a β sandwich, found in many immunoglobulin-like proteins, such as the human MCG protein (CATH code 4unuA00). The TIM barrel is a very common α/β fold, shown in an isomerase of the parasite *Leishmania mexicana* (CATH code 2vxnA00). The $\alpha - \beta$ plait fold is a two-layer sandwich, shown here in a human nucleoprotein, HNRNP (CATH code 1l3ka01). The Rossmann fold is a very common three-layer sandwich fold in the mixed $\alpha - \beta$ class, found, for example, in the D-amino acid oxidase from *Rhodotorula toruloides* (CATH code 1c0pA01). All images were generated using Pymol <http://www.pymol.org>

nearest neighbor in the training set. Here the nearest neighbor is found by computing first the mean normalized distance between the test sequence and all sequences in the training set that belongs to a given group, for all groups in the training set, and then taking the smallest of those mean distances. The results are gathered in a confusion matrix, C , whose element $C(i, j)$ corresponds to the number of test sequences from group i that have been classified as belonging to group j . The efficiency of the classifier d is then defined to be the quotient of the trace of the confusion matrix over the sum of all its elements. This quotient corresponds to the percentage of correctly classified sequences.

Protein structure comparison

We have used STRUCTAL [47] to perform geometric alignments of two curves representing two protein structures. STRUCTAL starts with an initial alignment (a correspondence between residues of the two structures), and computes the rigid-body transformation that leads to the “best” geometric superimposition of the corresponding residues. It then identifies an optimal alignment for this superposition, using dynamic programming. The new alignment defines a new correspondence, which is used to superimpose the structures again. This procedure is then iterated until it converges to a local optimum that depends on the initial alignment. To alleviate biases due to that dependence, STRUCTAL repeats the iterative search using several different initial correspondences.

The traditional measure of similarity between two protein structures after optimal alignment is the root mean

square displacement of atomic positions, also called cRMS for coordinate root mean square displacement, computed as:

$$cRMS = \sqrt{\frac{\sum_{i=1}^N \text{dist}(a_i, b_i)}{N}} \quad (8)$$

where N is the number of positions in the correspondence, and a_i and b_i are the Cartesian positions of two residues a and b from the two structures. The cRMS however is not a good measure of structural similarity [48]. Intuitively, a “good” measure of geometric, or structural similarity should favor alignments with many residues that are matched, low cRMS deviations, and few gaps. Unfortunately, these properties are not unrelated. For example, a lower cRMS deviation can always be achieved by selecting a smaller number of matches. In fact, given the fixed inter-CA distance there is the extreme case of alignments that only include two residues that have cRMS deviations of exactly zero. Also, the addition of gaps may lengthen the alignment without increasing the cRMS value. Different similarity measures have been proposed that attempt to balance these values in different ways. In this work, we have implemented the Structural Alignment Score, SAS:

$$SAS = 100 \frac{cRMS}{N}, \quad (9)$$

originally introduced by the authors of STRUCTAL [47].

Reproducibility

The program SeqKernel described above, as well as all the datasets of sequences used in this study are available from the authors upon request.

Results and Discussion

Two proteins with highly similar sequences almost always have similar structures. The reverse, however, is not always true. In a comprehensive analysis of the relationship between sequence similarity and structure similarity, Rost [16] has shown that pairs of proteins with similar structures possess, on average, only 8-10% sequence identity. This observation is at the root of the difficulties observed when attempting to classify proteins based on sequence information only. We have tested an alternative method to standard pairwise sequence comparison. We propose to use a string kernel that provides an alignment-free measure of the similarity of two protein sequences. We then use that measure to classify those protein sequences and compare the corresponding classification results with classifications derived from 3D structures and sequences only. Our aim is to parameterize the string kernel such that it performs better than sequence alignment based methods, mimicking to some extent the classifications derived from structure. We use CATH2833 and CATH793 as our main test sets. CATH2833 is a dataset of 2833 protein sequences that correspond to the three main classes of CATH: one α fold, one β fold, and three α/β folds (see “Methods” section above). CATH2833 was designed in such a way that the sequences of any pair of proteins included have statistically no detectable similarity, by enforcing a FASTA [43] SSEARCH E-value $> 10^{-4}$. CATH793 is a subset of CATH2833 that contains those proteins whose lengths are limited to a small range, from 120 to 180 amino acids.

Assessing the different protein sequence distances using receiver operator curve (ROC) analyses

ROC analysis of protein fold recognition based on FASTA E-values for pairwise sequence comparison, SAS scores for 3D structure comparison using STRUCTAL (see “Methods” above), and two conditions for the alignment-free SeqKernel comparisons are shown in panels a and b of Fig. 2 for the datasets CATH2833 and CATH793, respectively. These ROC analyses are based on averaged behaviors over the five folds included in those two datasets (i.e. a pair of sequences is considered to be positive if they belong to the same fold, independent of this fold. Results broken down by fold are given in Tables 2 and 3 for CATH2833 and CATH793, respectively.

The FASTA SSEARCH tool [43] is the implementation of a fast Smith and Waterman sequence comparison; it measures the similarity between two sequences either using directly the raw score of the alignment, or with a corresponding E-value. We have used the latter as a similarity measure. All SSEARCH alignments were performed using the BLOSUM62 substitution matrix [42], with gap penalties of -11 (opening) and -1 (extension). The ROC curves for this FASTA measure are marginally above the first diagonal, with AUC scores of 0.54 for both CATH2833 and CATH793. This behavior is expected, as by design all protein pairs in those datasets have little or no sequence similarity. The AUC values for the individual folds are very similar, ranging from 0.51 to 0.58, with one exception, fold 1.10.10 within the CATH793 dataset. We note however that this fold only contains 36 representatives in this dataset, making it the smallest set of all folds considered here. As such, this result is statistically unreliable.

Assignment of structural fold is expected to work best when it is based on 3D structural information. Indeed,

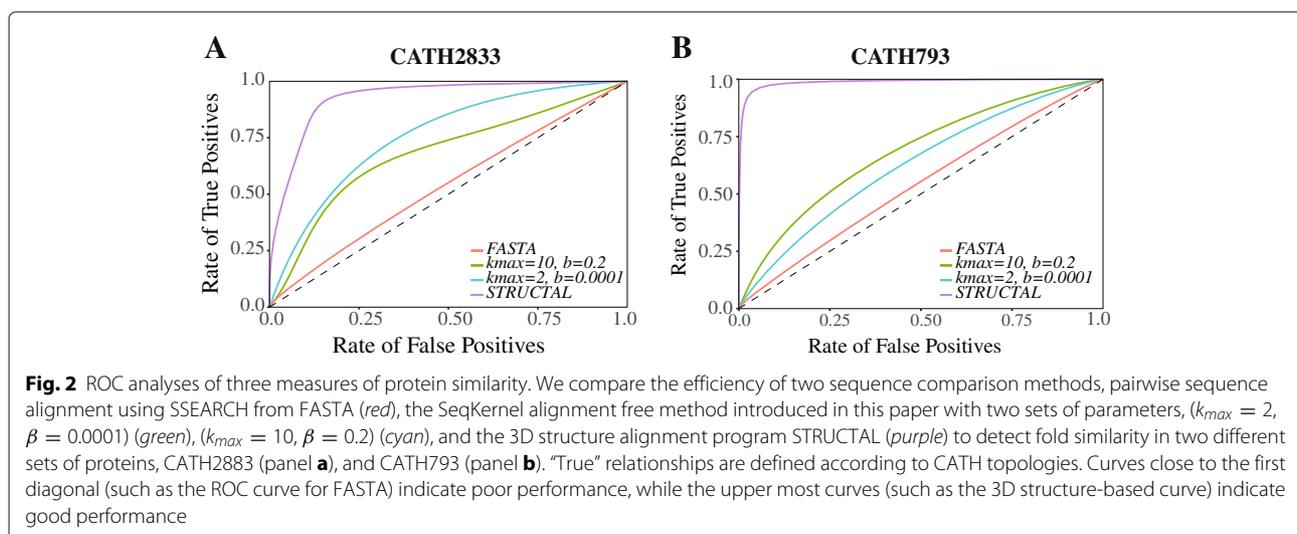


Table 2 Area Under the Curve (AUC) of ROC analyses for the FASTA, STRUCTAL, and SeqKernel distances for comparing protein sequences from CATH2833

CATH fold ID	N ^a	Distance			
		FASTA ^b	STRUCTAL ^c	SeKernel(0.2,10) ^d	SeqKernel(0.0001,2) ^e
1.10.10	381	0.58	0.85	0.31	0.62
2.60.40	555	0.58	0.97	0.38	0.65
3.20.20	251	0.53	0.99	0.88	0.85
3.30.70	368	0.53	0.92	0.33	0.62
3.40.50	1278	0.53	0.92	0.77	0.77
All five folds	2833	0.54	0.93	0.69	0.76

^aNumber of proteins in the fold^bAUC based on FASTA E-value^cAUC based on STRUCTAL SAS score^dand ^eAUC based on SeqKernel distance, with $(\beta, k_{max}) = (0.2, 10)$ (d) and $(\beta, k_{max}) = (0.0001, 2)$ (e)

the ROC curves obtained based on the SAS STRUCTAL scores illustrate excellent classification results, with AUC scores of 0.93 and 0.99 for CATH2833 and CATH793, respectively. We note that even with X-ray structure information the classification is not perfect, especially for CATH2833. This again is not a surprise. CATH is a semi-automatic classification of protein structures and some proteins are included in the same class based on more information than structure alone [37]. As such, two proteins may belong to the same class even though their structures are loosely similar. In addition, structural alignment programs work with heuristic algorithms and as such may miss the optimal alignment [48]. Finally, it is possible that a small fully α or fully β protein is found to be similar to an α/β protein, based on local alignment of the helical, or strand regions of the proteins. This is observed for example for proteins in fold 1.10.10 (the arc repressor all- α fold), with an AUC of 0.85 for CATH2833. That said, STRUCTAL scores based on X-ray structures still perform remarkably well.

The ROC curves based on the alignment free sequence comparisons obtained with SeqKernel are intermediate

between the FASTA and STRUCTAL curves, for the two sets of parameters considered. Clearly, kernel similarity measures improve the classification of proteins into folds, especially for those proteins whose sequences bear little similarity. This improvement is significant, from an overall AUC score of 0.54 for CATH2833 with FASTA to AUC values between 0.63 and 0.76 for CATH2833 with SeqKernel, depending on the values given to the parameters k_{max} and β . Similar improvements are observed for CATH793. Interestingly, the improvements between FASTA and SeqKernel are not consistent over all types of folds considered. SeqKernel performs significantly better for proteins in the $\alpha - \beta$ class (folds 3.20.20 and 3.40.50) than for the full α fold (1.10.10) and for the full β fold (2.60.40). Of similar interest, there is a discrepancy between the results obtained on CATH2833 and CATH793: for the former, a significantly better overall AUC is observed when the pair (k_{max}, β) is set to (2,0.0001), while for the latter, a better overall AUC is observed when the pair (k_{max}, β) is set to (10,0.2). In addition, results using $(k_{max}, \beta)=(2,0.0001)$ are more consistent over the five folds considered. While differences

Table 3 Area Under the Curve (AUC) of ROC analyses for the FASTA, STRUCTAL, and SeqKernel distances for comparing protein sequences from CATH793

CATH fold ID	N ^a	Distance			
		FASTA ^b	STRUCTAL ^c	SeKernel(0.2,10) ^d	SeqKernel(0.0001,2) ^e
1.10.10	36	0.75	0.91	0.61	0.61
2.60.40	130	0.55	0.98	0.44	0.54
3.30.70	52	0.51	0.89	0.45	0.47
3.40.50	573	0.57	0.98	0.69	0.62
All five folds	791	0.55	0.98	0.69	0.62

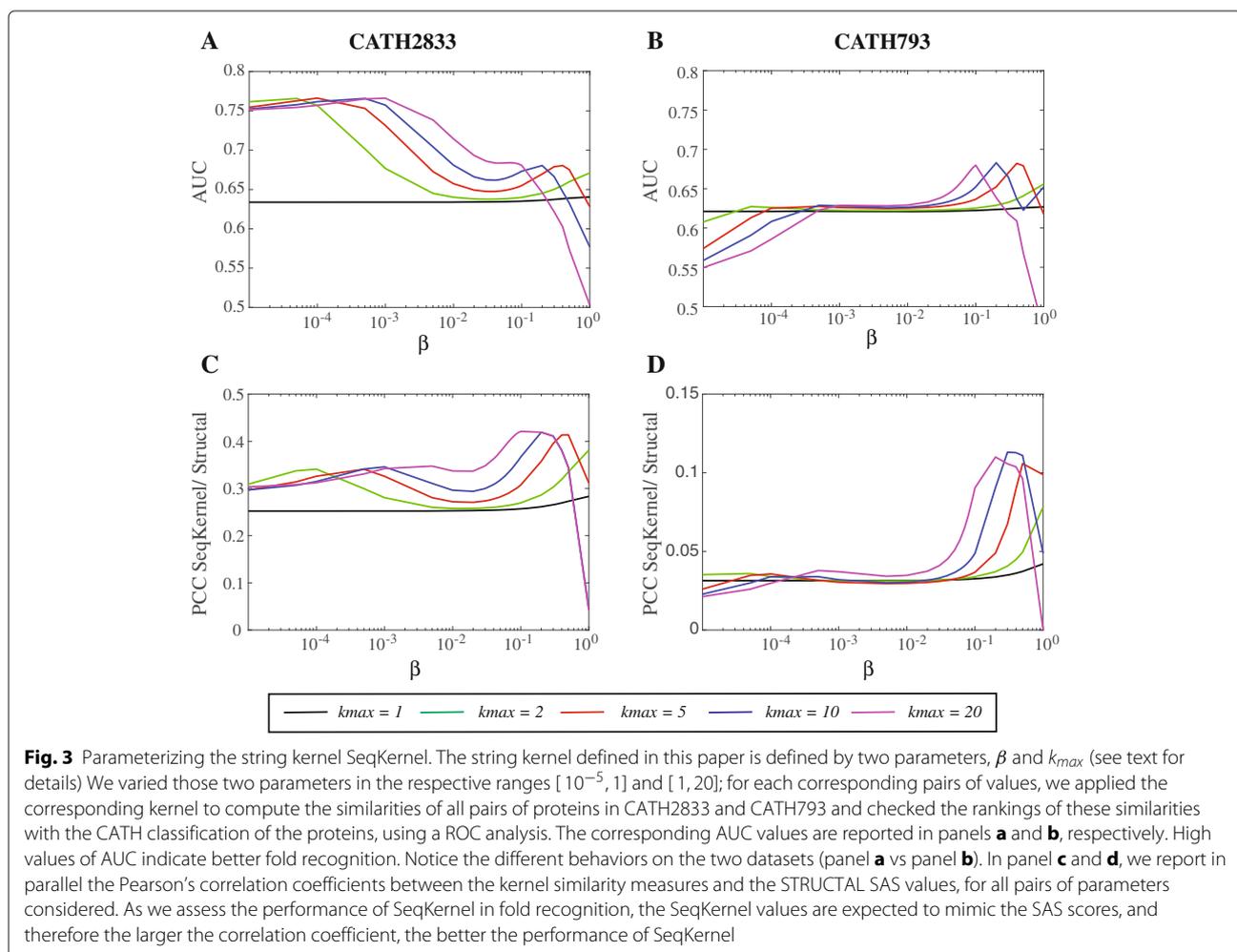
^aNumber of proteins in the fold^bAUC based on FASTA E-value^cAUC based on STRUCTAL SAS score^dand ^eAUC based on SeqKernel distance, with $(\beta, k_{max}) = (0.2, 10)$ (d) and $(\beta, k_{max}) = (0.0001, 2)$ (e)

are expected for different parameter values, the ranking of those parameter values is expected to remain the same on different datasets, especially here as CATH793 is a subset of CATH2833. This difference warrants a systematic study of the impact of the kernel parameters on its performance in fold recognition.

Understanding the parameters β and k_{max} for the string kernel SeqKernel

The string kernel, SeqKernel, considered in this paper depends on two parameters, the coefficients β and k_{max} . β is a power coefficient, i.e. it is used to compute the Hadamard power of the input substitution matrix SM . As it is used in a Hadamard power, all strictly positive values for β are possible, as the power matrix remains a Gram matrix. The parameter k_{max} defines the longest k -mers that are considered in the comparison of the two sequences. Note that the maximum value for k_{max} is the size of the smallest sequence. Intuitively, using large values for k_{max} are expected to help as the longer k -mers capture correlations in the protein sequences. However,

as described in the implementation section, large k_{max} values come at a computational cost. We have tested a range of values for β from very small, 10^{-5} , to relatively large, 1, and a range of values for k_{max} , from 1 (i.e. single amino acid comparison) to 20. For pairs of values (β , k_{max}) taken from their respective ranges, we computed the similarity scores for all pairs of proteins in CATH2833 and CATH793, and assessed the ability of those scores for fold recognition using ROC analysis. The resulting AUC scores are reported in Fig. 3. Note that the higher the AUC, the better the performance. In parallel, we also compared the string kernel similarity scores with the corresponding STRUCTAL SAS scores. As STRUCTAL was found to perform extremely well in the ROC analysis (see Fig. 2), the SeqKernel similarity scores should mimic the SAS scores for good fold recognition. We therefore computed the Pearson's correlation coefficient PCC between the two sets of values, i.e. similarity scores and SAS values, for all pairs of proteins that belong to the same folds. The resulting PCC values are also reported in Fig. 3. Again, the higher the PCC, the better the performance.



The results on the CATH2833 dataset show two different behaviors depending on the β values: for very small β values (below 10^{-3}), all the $AUC=f(k_{max})$ curves reach the same plateau with a relatively high value of 0.76, while for larger values of β (> 0.1), the same curves behave similarly but reach different maxima, with the β values corresponding to these maxima decreasing as k_{max} increases. Interestingly, the results on the CATH793 dataset are very different: while the same behavior is observed for the large values of β , SeqKernel provides poor fold recognition for very small β values. The key difference between the two datasets is that the latter includes proteins that cover a very small range of lengths. The discrepancy in behavior therefore hints to SeqKernel being able to pick differences in protein lengths for small β values. Indeed, as $\beta \rightarrow 0$, all the values in the kernel K_1 tend to 1, and K_1 becomes a matrix of ones. While this matrix does not correspond to a kernel (it is not positive definite), it can still be used in practice as input to SeqKernel. Using this matrix, the semi-kernel K_3 can in fact be computed analytically:

$$K_3(S, T) = \sum_{k=1}^{k_{max}} (n - k + 1)(m - k + 1) \quad (10)$$

Using this semi-kernel, we found that it performs with an AUC of 0.76 on the classification of the proteins in CATH2833. This high value of 0.76 therefore only reflects the differences in the distribution of lengths of the proteins in the five folds. Those differences are significant, see Table 1. While an interesting observation by itself, this is not the string kernel we are interested in. Indeed, Eq. 10 shows that it is independent of the actual sequences themselves, and only captures length differences; for example, the corresponding correlation semi-kernel takes the value of 1 for all pairs of sequences of the same lengths. These results suggest to use a kernel based on larger values of β . Figure 3a and b indicate that any value of k_{max} is possible, pending that the proper value for β is chosen. We suggest using the pair $(\beta, k_{max})=(0.2,10)$. Interestingly, the corresponding kernel leads to the same AUCs on the two datasets, CATH2833 and CATH793 (see above).

Comparisons of the kernel values with the STRUCTAL SAS values confirm this analysis, see panels c and d in Fig. 3. The PCC values between these two measures are found low for small values of β . In such conditions, the kernel was shown to mainly capture lengths, while the STRUCTAL scores are mostly independent of lengths. For larger values of β , the PCC values reach maxima for the same pairs of values (β, k_{max}) than the AUC values.

Choosing the input substitution matrix

There is one other parameter that defines the string kernel SeqKernel, namely the input substitution matrix. We compared fold recognition performance as measured by

ROC analysis of both FASTA and SeqKernel (with β and k_{max} set to 0.2 and 10, respectively) on the two datasets CATH2833 and CATH793 for different input substitution matrices. Results are shown in Fig. 4. As a reminder, FASTA and SeqKernel use different matrices as input: FASTA uses a BLOSUM-like matrix, while SeqKernel uses its own count-based matrix. Those two matrices however are mathematically related by a simple *log* function. We note that there are little variations in the performance of both FASTA and SeqKernel as we change their input matrices. More surprisingly, SeqKernel performs quite well with the Identity matrix as input. This indicates that at least for the two datasets considered here, a strict score that does not favor replacement is good enough for fold recognition.

Protein fold classification

We extended the ROC analyses of our sequence similarity measures to assessing their performances for actual fold recognition by performing a set of computational fold classification experiments. Note that we are not interested in establishing a classification for a family of protein structures: rather, we are interested in testing the ability of distance measures based on sequences to assign a protein to the structural class it belongs to. We perform this test using a series of computational experiments. For those experiments, we consider all folds included in Cath40e4 with at list 40 members, excluding the five folds considered in CATH2833, to eliminate possible biases as SeqKernel was parametrized based on these folds. There are 40 such folds, for a total of 3744 proteins, which we

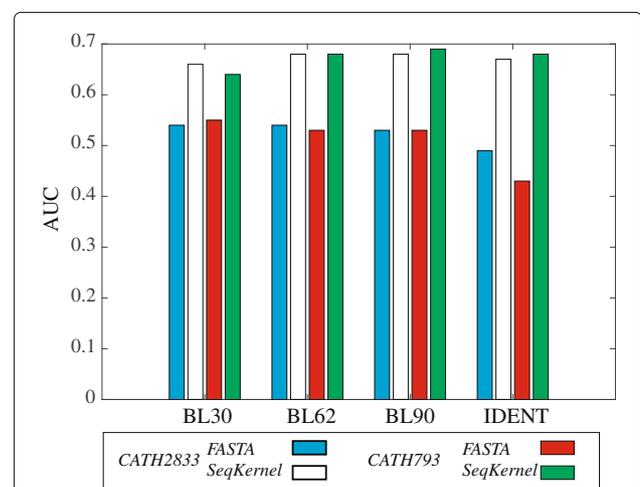


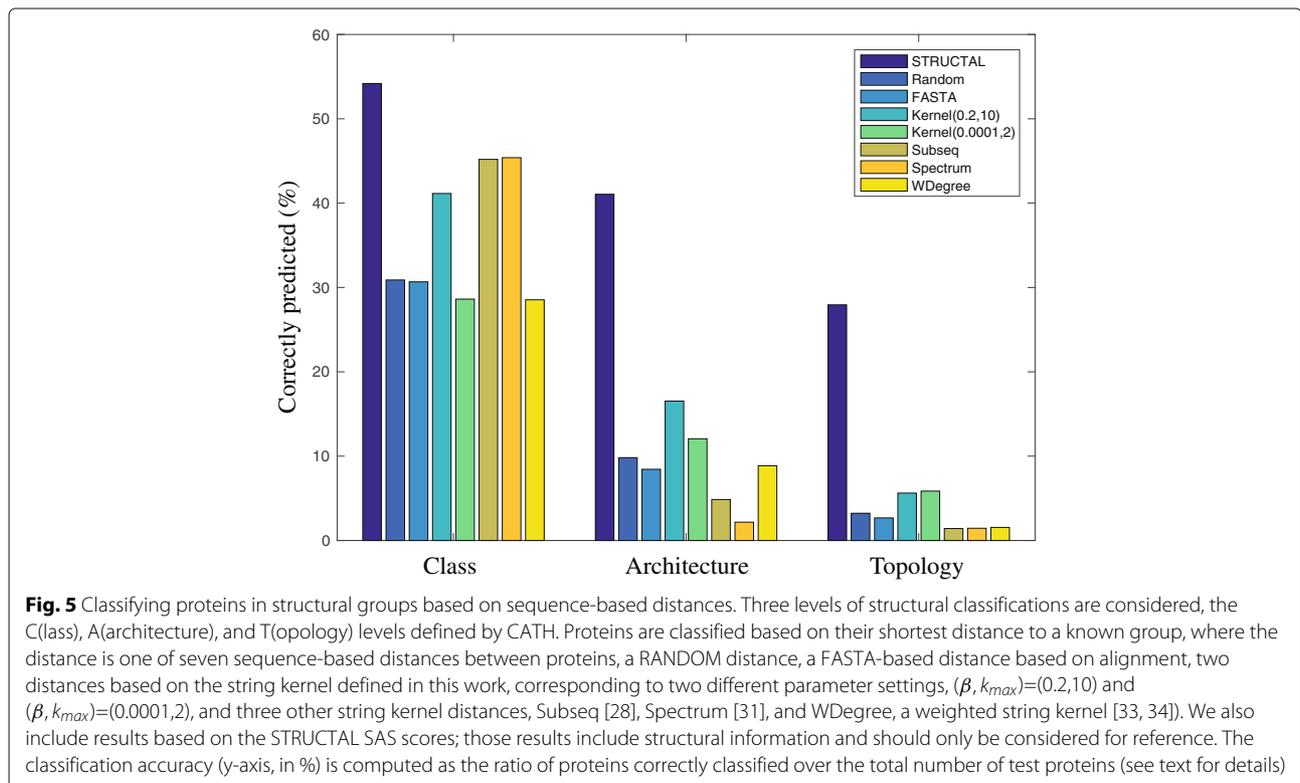
Fig. 4 Choosing the substitution matrix. Comparison of the performances of FASTA and SeqKernel ($\beta, k_{max})=(0.2,10)$, as measured by AUC values, for different input substitution matrices, namely the BLOSUM30 (BL30), BLOSUM62 (BL62), BLOSUM90 (BL90), and the Identity matrices. The two values for FASTA and for Seqkernel correspond to the two datasets CATH2833 and CATH793, respectively

refer to as CATH3744. We consider three levels of classifications for those proteins, as defined by CATH: the Class level (3 such classes, α , β , and mixed α/β , the Architecture level (13 such architectures, 3 in the α class, 4 in the β class, and 6 in the mixed α/β class), and at the Topology (or fold) level (40 such topologies, 10 in the α class, 10 in the β class, and 20 in the mixed α/β class). Each experiment starts with those proteins, and a choice for the classification level. We pick at random half of the proteins of each group at the level considered, and define them as the *training set*. The remaining proteins form the *test set*. The training set is used to define the groups for the distance-based classifiers (see “Methods” for details). The experiment then proceeds by assigning a group to each protein in the test set. A protein is considered successfully classified if it is assigned to its actual CATH group (C, A, or T, based on the experiment considered). Experiments are repeated 10,000 times and the results averaged to remove potential bias in the choice of proteins that define the folds.

For those fold classification experiments, we considered 8 possible distances between protein sequences. As a null reference, we assign a random value between 0 and 1 for the distance between two proteins. We call this distance RANDOM. On the other end of the spectrum, we include results based on the STRUCTAL SAS scores; those results directly reflect structural similarities. The third distance is based on the FASTA SSEARCH E-value. The fourth and

fifth distances are computed with SeqKernel. As described in the previous section, there seem to be two regimes for SeqKernel, based on the values for the parameters β and k_{max} . We picked one example from each regime, setting (β, k_{max}) to (0.0001,2) and (0.2,10), respectively. The last three distances are based on other string kernels for comparing strings. We considered the subsequence string kernel introduced by Lodhi et al [28], Subseq, the Spectrum string kernel of Leslie et al [31], and WDegree, the weighted string kernel of Ratsch and colleagues [33, 34]. For those last kernel distances, we used the implementations provided in the package Harry [49, 50], with default parameters. Results of the classification experiments are given in Fig. 5.

Results from the reference RANDOM distances are on par with what is expected from the numbers of groups at each level of classification considered: 31% at the C level ($1/3 \approx 33\%$), 9% at the A level ($1/13 \approx 8\%$), and 3% at the T level ($1/40 \approx 3\%$). With no surprise, the results based on FASTA mirror those RANDOM results: as described above, there is no information in the FASTA distances by design. Also, without surprise, the classifications based on the STRUCTAL distances between structures are significantly more accurate than those based on sequence only. It is interesting however that even those results are far from perfect. At the C level for example, only 54% of the proteins are correctly classified. Most misclassification come from $\alpha + \beta$ proteins being classified as α or β



proteins. This is not unexpected, as STRUCTAL performs a local structural alignment. The classifications obtained with SeqKernel with (β, k_{max}) set to (0.2,10) are significantly more accurate than those observed with the other sequence-based distances, at least at the Architecture and Topology levels. This includes the results obtained with SeqKernel with (β, k_{max}) set to (0.0001,2), confirming that a somewhat large value for β is preferred. Interestingly, the Subseq and Spectrum string kernels perform better at the Class level. It is likely that those two kernels capture the amino acid compositions of the sequences, which have been shown to be good discriminants of the protein structural class (as defined by CATH) [51, 52].

Alignment-free phylogeny reconstruction

A large number of biological modeling methods depend on accurate (multiple) sequence alignments. Protein structure prediction is one of those methods; it relies heavily on fold recognition. In the previous sections, we covered the use of alignment free methods for performing this task, comparing our string kernel SeqKernel with alignment-based methods. Another important modeling task that relies on sequence alignment is phylogenetic tree inference, a critical step in evolutionary studies. The majority of methods that perform this task follow a two step process, with the construction of a multiple sequence alignment followed by statistical tree inference [53, 54]. These methods, though widely used, have known limitations related to uncertainties in the multiple sequence alignment [38, 39, 55–57]. Among all the approaches recently developed to alleviate those limitations, we note the alignment-free methods based on analyses of k -mers in the sequences that are compared [39, 40, 58, 59]. As SeqKernel provides a mean to quantify the similarity between two sequences using such k -mers, we tested it on a simple toy problem of phylogenetic tree inference originally described by Thorne and Kishino [38].

To illustrate their methods for computing evolutionary distances between protein sequences, they considered 10 sequences of the second largest RNA polymerase subunit. We have repeated their analysis on the same sequences. These sequences include two eukaryotic pol I sequences, two eukaryotic pol II sequences, two eukaryotic pol III sequences, two archaeobacterial sequences, a eubacterial sequence, and a chloroplast sequence; they are defined in Table 4.

We considered 5 different distance matrices between those sequences. The first matrix is the original distance matrix proposed by Thorne and Kishino. The second matrix is derived from the multiple sequence alignment of those 10 sequences, computed with Clustal Omega [60], and converted into a distance matrix using the program ProtDist from the software package Phylip [61]. We used the Jones, Taylor and Thornton model of amino acid change [62] within ProtDist to compute the distances. The third matrix is based on the Bit scores of the sequence alignments generated by SSEARCH from FASTA, computed with Blosum62 as a substitution matrix and gap penalties of -11/-1 for opening/extension, respectively. The fourth and fifth matrices are derived from SeqKernel, with two settings for the parameters β and k_{max} , namely (0.0001,2) and (0.2,10). We have built trees based on those five distance matrices using the programs Fitch and Drawtree from the Phylip package [61]. Fitch is an implementation of the Fitch and Margoliash [63] method for constructing trees from a distance matrix under the “additive tree model”. In this model, the distance between two sequences is expected to be equal to the sum of branch lengths between the sequences on the tree. These five trees are shown in Fig. 6.

With the exception of the tree generated from the distance matrix derived from the SeqKernel distances with β and k_{max} set to (0.0001,2) the trees are qualitatively similar to each other. As the 10 sequences have approximately

Table 4 The ten RNA polymerases

Abbreviation	Protein	Species	Accession number	Length
SC1	RNA Pol I subunit RPA2	<i>Yeast</i>	RPA2_YEAST	1203
DR1	RNA Pol I subunit RPA2	<i>Drosophila megalonaster</i>	RPA2_DROME	1129
SC2	RNA Pol II subunit RPB2	<i>Yeast</i>	RPB2_YEAST	1203
DR2	RNA Pol II subunit RPB2	<i>Drosophila megalonaster</i>	RPB2_DROME	1129
SC3	RNA Pol III subunit RPC2	<i>Yeast</i>	RPC2_YEAST	1203
DR3	RNA Pol III subunit RPC2	<i>Drosophila megalonaster</i>	RPC2_DROME	1129
SUL	RNA Pol subunit β	<i>Sulfolobus acidocaldarius</i>	A0A0U3H235_9CREN	1126
MET	RNA Pol subunit B'	<i>Methano-bacterium thermoautotrophicum</i>	RPOB1_METTW	1123
ESC	RNA Pol subunit β	<i>Escherichia coli</i>	RPOB_ECOSE	1342
SPI	Chloroplast RNA Pol subunit β	<i>Spinacia oleracea</i>	RPOB_SPIOL	1070

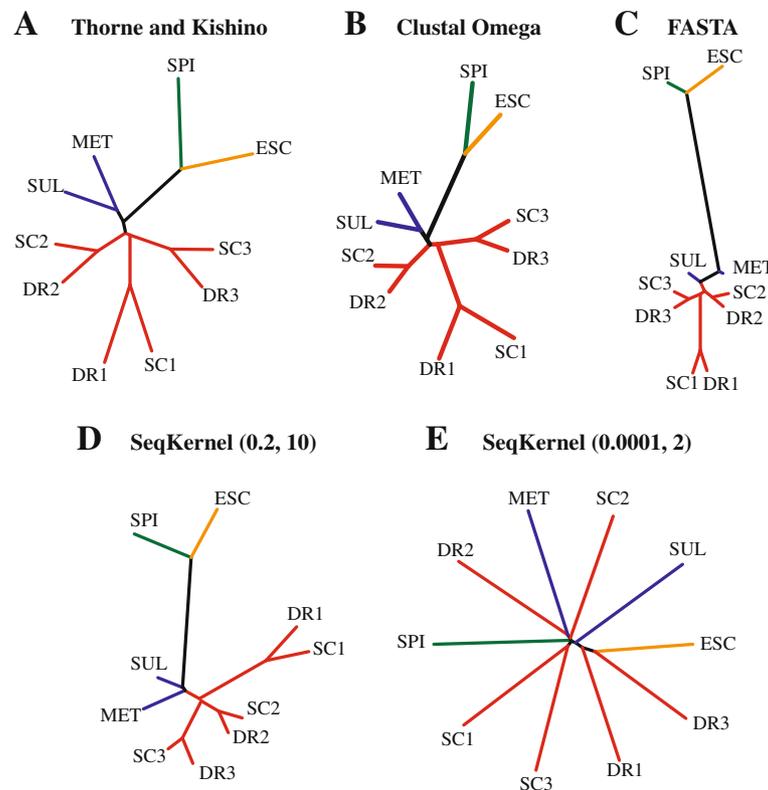


Fig. 6 Inferred RNA polymerase phylogeny. Five different phylogenetic trees were constructed for a group of 10 RNA polymerases, using five different distance measures between their sequences: **(a)** the original distances proposed by Thorne and Kishino [38], **(b)** distances derived from a Multiple Sequence Alignment of the 10 sequences computed with Clustal Omega [60], **(c)** the BIT distances from FASTA SSEARCH tools, and the distances computed with SeqKernel with two different settings, namely $(\beta, k_{max}) = (0.2, 10)$ **(d)** and $(\beta, k_{max}) = (0.0001, 2)$ **(e)**. Abbreviations for species names are provided in Table 4. Branches corresponding to eukaryotic sequences are colored red, those for archaeobacterial sequences are shown in blue, and the eubacterial and chloroplast sequences are highlighted in orange and green, respectively

the same lengths, and as SeqKernel with a small value of β basically captures differences in length, the exception is not surprising. This result however does reinforce our choice for (β, k_{max}) to $(0.2, 10)$. To quantify the differences between the four remaining trees, we first scaled the distance matrices so that all distances ranged between 0 and 1, and then regenerated the trees. The resulting trees are then compared with the program TreeDist, implemented in the software package Phylip. TreeDist is based on the branch score distance of Kuhner and Felsenstein [64] to evaluate the similarity between two trees. We find that the distances between the original tree of Thorne and Kishino and the Clustal-based, FASTA-based, and SeqKernel-based trees are 0.3, 0.57, and 0.39, respectively. Note that the distance between the original tree and the SeqKernel-based tree when (β, k_{max}) is set to $(0.0001, 2)$ is 0.99. While we cannot assess the meaning of the absolute values of these distances, and the significance of the differences between those values, we do notice that the original Thorne and Kishino tree resembles most the

tree computed with the Clustal-based tree, and the tree computed with the method introduced here.

Conclusions

The amino acid sequence of a protein encodes for its structure and ultimately its function in a cell. As such, sequence comparison remains one of the core tools used in many modeling methods in molecular biology that handle problems such as protein fold recognition and phylogenetic tree inference. We have shown that a string kernel that captures the similarity of all k -mers in two protein sequences provides an alignment-free method for fold recognition and phylogenetic tree reconstruction that performs well when its parameters have been selected appropriately. We refer to this string kernel as SeqKernel. It depends on two parameters, β , a power coefficient that modulates the values of the input substitution matrix, and k_{max} , the maximum lengths of k -mers considered. We have performed a systematic analysis of the impact of those two parameters on the performance of SeqKernel in

fold recognition experiments involving remote homologs. We have shown that on our test datasets, SeqKernel performs remarkably well for very small values of β ($< 10^{-3}$), independently of k_{max} . With such small values of β however, SeqKernel is basically tuned to capture the difference in lengths of proteins, which is not of interest for fold recognition. We have shown that for larger values of β , there are pairs of values (β, k_{max}) that provide significant performance in fold recognition. We suggest to use the pair $(\beta, k_{max})=(0.2,10)$. Interestingly, in the first presentation of a kernel equivalent to SeqKernel [36], the authors advocate the use of $k_{max} = 10$, and referred to β as a mysterious parameter that they fixed to 0.11, i.e. close to the value of 0.2 that we propose here.

SeqKernel, just like any alignment-based sequence comparison method, depends on a score matrix, also called substitution matrix. Such a matrix provides a quantitative measure of the similarities of amino acids. We have shown that SeqKernel is not sensitive to the choice of this score matrix, at least on the two datasets CATH2833 and CATH793 used in this study. We note however that most of the score matrices, such as the BLOSUM matrices considered in this study, have not been optimized for the purpose of fold recognition. There has already been attempts to perform such optimization [65, 66]. In future studies, we will explore further the problem of defining an optimal score matrix within the context of SeqKernel.

This paper represents work in progress. It is dedicated to the understanding of the parameters that define the string kernel SeqKernel as well as to the illustration of its applications in fold recognition problems and phylogenetic tree inference. In the latter, we have only presented a small toy example to highlight that a alignment-free estimate of the distances between protein sequences can perform as well as a method based on multiple sequence alignment. Much work remains to be done before SeqKernel can become a commodity tool for sequence analysis. The scores computed with SeqKernel, while corresponding to a metric, do not include information on significance. The computing time for comparing two sequences using SeqKernel is high, suggesting that SeqKernel may not be used for large database searches. All the tests performed in this study were related to single domain proteins, or to specific domains within a protein. There is a need to expand the range of applications of SeqKernel to multi-domain proteins, as those are more the norm than single-domain proteins. Applications to phylogenomics need to be expanded and validated on a much larger scale. All those points will serve as research topics in our future studies.

Abbreviations

AUC: Area under the curve; cRMS: Coordinate root mean square; ROC: Receiver operator curve

Acknowledgements

None.

Funding

Patrice Koehl acknowledges support from the Ministry of Education of Singapore through Grant Number: MOE2012-T3-1-008.

Availability of data and materials

The program SeqKernel described in the paper, as well as all the datasets of sequences used in this study are available from the authors upon request.

Authors' contributions

PK conceived the study, including the methodological choices and the selection of proteins, and wrote the programs required for the analyses. SN performed the computational experiments and carried out the analyses of the results. SN and PK wrote the paper as well as its subsequent revisions. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Author details

¹Biotechnology program, University of California, Davis, 1, Shields Avenue, 95616 Davis, CA, USA. ²Department of Computer Science and Genome Center, 1, Shields Avenue, 95616 Davis, CA, USA.

Received: 21 October 2016 Accepted: 23 February 2017

Published online: 28 February 2017

References

1. Oldfield CJ, Dunker AK. Intrinsically Disordered Proteins and Intrinsically Disordered Protein Regions. *Ann Rev Biochem.* 2014;83:553–84.
2. Dill KA, Ozkan SB, Weikl TR, Chodera JD, Voelz VA. The protein folding problem: when will it be solved?. *Curr Opin Struct Biol.* 2007;17:342–6.
3. Das R, Baker D. Macromolecular modeling with Rosetta. *Annu Rev Biochem.* 2008;77:363–82.
4. Bairoch A, Apweiler R. The Swiss-Prot protein sequence data bank and its supplement TrEMBL in. *Nucl Acids Res.* 2000;28:45–8.
5. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, et al. The Protein Data Bank. *Nucl Acids Res.* 2000;28:235–42.
6. The Uniprot Consortium. UniProt: a hub for protein information. *Nucl Acids Res.* 2015;43:D204–D212.
7. Waterman MS. Introduction to computational biology: maps, sequences, and genomes. Boca Raton: Chapman and Hall / CRC Interdisciplinary Statistics; 1995.
8. Durbin R, Eddy SR, Krogh A, Mitchison G. Biological sequence analysis: probabilistic models of nucleic acids and proteins. New York: Cambridge University Press; 1998.
9. Gusfield D. Algorithms on strings, trees, and sequences: computer science and computational biology. New York: Cambridge University Press; 1997.
10. Schwartz RM, Dayhoff MO. Matrices for detecting distant relationships. *Atlas of Protein Sequence and Structure.* 1978;5:345–52.
11. Henikoff S, Henikoff JG. Amino acid substitution matrices. *Adv Protein Chem.* 2000;54:73–97.
12. Ladunga I. PHYSEAN: PHYSical SEquence ANALYSIS for the identification of protein domains on the basis of physical and chemical properties of amino acids. *Bioinformatics.* 1999;15:1028–38.
13. Smith TF, Waterman MS. Identification of Common Molecular Subsequences. *J Mol Biol.* 1981;147:195–7.
14. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol.* 1970;48:443–53.
15. Storey JD, Sigmund D. Approximate p-values for local sequence alignments: numerical studies. *J Comput Biol.* 2001;8:549–56.
16. Rost B. Twilight zone for sequence alignment. *Protein Eng.* 1999;12:85–94.

17. Wallace IM, Blackshields G, Higgins DG. Multiple sequence alignments. *Curr Opin Struct Biol.* 2005;15:261–6.
18. Vinga S, Almeida J. Alignment-free sequence comparison—a review. *Bioinformatics.* 2003;19:513–23.
19. Bonham-Carter O, Steele J, Bastola D. Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis. *Brief Bioinform.* 2014;15:890–915.
20. Vinga S. Alignment-free methods in computational biology. *Brief Bioinform.* 2014;15:341–2.
21. Schwende I, Pham TD. Pattern recognition and probabilistic measures in alignment-free sequence analysis. *Brief Bioinform.* 2014;15:354–68.
22. Ulitsky I, Burstein D, Tuller T, Chor B. The average common substring approach to phylogenetic reconstruction. *J Comput Biol.* 2006;13:336–50.
23. Didier G, Corel E, Laprevotte I, Grossmann A, Landes-Devauchelle C. Variable length decoding and alignment-free sequence comparison. *Theor Comput Sci.* 2012;462:1–11.
24. Ma B, Tromp J, Li M. PatternHunter: faster and more sensitive homology search. *Bioinformatics.* 2002;18:440–5.
25. Burkhardt S, Kärkkäinen J. Better filtering with gapped q-grams. *Fundam Inf.* 2003;56:51–70.
26. Keich U, Li M, Ma B, Tromp J. On spaced seeds for similarity search. *Discrete Appl Math.* 2004;138:253–63.
27. Leimeister CA, Boden M, Horwege S, Lindner S, Morgenstern B. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics.* 2014;15:1991–9.
28. Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins C. Text classification using string kernels. *J Mach Learn Res.* 2002;2:419–44.
29. Diekhans TJM, Haussler D. A discriminative framework for detecting remote protein homologies. *J Comput Biol.* 2000;7:95–114.
30. Liao L, Noble WS. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *J Comput Biol.* 2003;10:857–68.
31. Leslie C, Eskin E, Noble WS. The spectrum kernel: A string kernel for SVM protein classification. In: Altman RB, Dunker AK, Hunter L, Lauderdale K, Klein TE, editors. *Proceedings of the Pacific Symposium on Biocomputing; Kauai*: 2002. p. 564–75.
32. Leslie CS, Eskin E, Cohen A, Weston J, Noble WS. Mismatch string kernels for discriminative protein classification. *Bioinformatics.* 2004;20:467–76.
33. Rättsch G, Sonnenburg S, Schölkopf B. RASE: Recognition of alternatively spliced exons in *C. elegans*. *Bioinformatics.* 2005;21:i369–i377.
34. Ben-Hur A, Ong CS, Sonnenburg S, Schölkopf B, Rättsch M. Support Vector Machines and Kernels for Computational Biology. *PLoS Comput Biol.* 2008;4:e1000173.
35. Saigo H, Vert JP, Ueda N, Akutsu T. Protein homology detection using string alignment kernels. *Bioinformatics.* 2004;20:1682–9.
36. Shen WJ, Wong HS, Xiao QW, Guo X, Smale S. Introduction to the Peptide Binding Problem of Computational Immunology: New Results. *Found Comput Math.* 2013;14:951–84.
37. Sillitoe I, Lewis TE, Cuff AL, Das S, Ashford P, Dawson NL, et al. CATH: comprehensive structural and functional annotations for genome sequences. *Nucl Acids Res.* 2015;43:D376–D381.
38. Thorne JL, Kishino H. Freeing phylogenies from artifacts of alignment. *Mol Biol Evol.* 1992;9:1148–62.
39. Chan CX, Ragan MA. Next-generation phylogenomics. *Biology direct.* 2013;8:3.
40. Chan CX, Bernard G, Poirion O, Hogan JM, Ragan MA. Inferring phylogenies of evolving sequences without multiple sequence alignment. *Scientific Reports.* 2014;4:6504.
41. Haussler D. *Convolution Kernels on Discrete Structures*: University of California, Santa Cruz; 1999. UCS-CRL-99-10.
42. Henikoff S, Henikoff J. Amino Acid Substitution Matrices from Protein Blocks. *Proc Natl Acad Sci (USA).* 1992;89:10915–9.
43. Pearson W, Lipman D. Improved tools for biological sequence comparison. *Proc Natl Acad Sci (USA).* 1988;85:2444–8.
44. Le Q, Pollastri G, Koehl P. Structural Alphabets for Protein Structure Classification: A Comparison Study. *J Mol Biol.* 2009;387:431–50.
45. Li J, Koehl P. 3D representations of amino acids - applications to protein sequence comparison and classification. *Comp Struct Biotech J.* 2014;11:47–58.
46. Gribskov M, Robinson NL. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Comput Chem.* 1996;20:25–33.
47. Subbiah S, Laurents DV, Levitt M. Structural similarity of DNA-binding domains of bacteriophage repressors and the globin fold. *Curr Biol.* 1993;3:141–8.
48. Kolodny R, Koehl P, Levitt M. Comprehensive evaluation of protein structure alignment methods: scoring by geometric measures. *J Mol Biol.* 2005;346:1173–88.
49. Rieck K. Similarity measures for sequential data. *WIREs: Data Mininf and Knowledge Discovery.* 2011;1:296–304.
50. Rieck K, Wresnegger C. Harry: a tool for measuring string similarity. *J Mach Learn Res.* 2016;17:1–5.
51. Chou KC. A novel approach to predicting protein structural classes in a (20-1)-D amino acid composition space. *Proteins: Struct Func Genet.* 1995;21:319–44.
52. Xiao X, Lin WZ, Chou KC. Using grey dynamic modeling and pseudo amino acid composition to predict protein structural classes. *J Comp Chem.* 2008;29:2018–24.
53. Felsenstein J. Phylogenies from molecular sequences: inference and reliability. *Annu Rev Genet.* 1988;22:521–65.
54. Chatzou M, Magis C, Chang JM, Kemena C, Bussotti G, Erb I, et al. Multiple sequence alignment modeling: methods and applications. *Brief Bioinform.* 2015;17:1009–23.
55. Höhl M, Ragan MA. Is multiple-sequence alignment required for accurate inference of phylogeny?. *Syst Biol.* 2007;56:206–21.
56. Wong KM, Suchard MA, Huelsenbeck JP. Alignment uncertainty and genomic analysis. *Science.* 2008;319:473–6.
57. Wu MT, Chatterji S, Eisen JA. Accounting for alignment uncertainty in phylogenomics. *PLoS ONE.* 2012;7:e30288.
58. Haubold B. Alignment-free phylogenetics and population genetics. *Brief Bioinform.* 2014;15:407–18.
59. Bogusz M, Whelan S. *Phylogenetic Tree Estimation With and Without Alignment: New Distance Methods and Benchmarking.* *Syst Biol.* 2016.
60. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol Syst Biol.* 2011;7:539.
61. Felsenstein J. PHYLIP – Phylogeny Inference Package (Version 3.2). *Cladistics.* 1989;5:164–6.
62. Jones DT, Taylor WR, Thornton JM. The rapid generation of mutation data matrices from protein sequences. *CABIOS.* 1992;8:275–82.
63. Fitch WM, Margoliash E. construction of phylogenetic trees. *Science.* 1967;155:279–84.
64. Kuhner MK, Felsenstein J. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol Biol Evol.* 1994;11:459–68.
65. Kann M, Qian B, Goldstein RA. Optimization of a new score function for the detection of remote homologs. *Proteins: Struct Func Genet.* 2000;41:498–503.
66. Saigo H, Vert JP, Akutsu T. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC Bioinformatics.* 2006;7:246.

Submit your next manuscript to BioMed Central and we will help you at every step:

- We accept pre-submission inquiries
- Our selector tool helps you to find the most relevant journal
- We provide round the clock customer support
- Convenient online submission
- Thorough peer review
- Inclusion in PubMed and all major indexing services
- Maximum visibility for your research

Submit your manuscript at
www.biomedcentral.com/submit

