Designing Antimicrobial Peptides with Weighted Finite-State Transducers

Christopher Whelan, Brian Roark, and Kemal Sönmez

***Abstract*— The design of novel antimicrobial peptides (AMPs) is an important problem given the rise of drug-resistant bacteria. However, the large size of the sequence search space, combined with the time required to experimentally test or simulate AMPs at the molecular level makes computational approaches based on sequence analysis attractive. We propose a method for designing novel AMPs based on learning from *n*-gram counts of classes of amino acid residues, and then using weighted finite-state machines to produce sequences that incorporate those features that are strongly associated with AMP sequences. Finite-state machines are able to generate sequences that include desired *n*-gram features. We use this approach to generate candidate novel AMPs, which we test using third-party prediction servers. We demonstrate that our framework is capable of producing large numbers of novel peptide sequences that share features with known antimicrobial peptides.**

I. INTRODUCTION

Antimicrobial peptides (AMPs) have attracted considerable attention as a potential new source of therapeutic agents effective against microorganisms that have developed resistance to traditional drugs [1]. However, the size of the search space makes it difficult to discover new AMPs through experimental testing; for a typical AMP sequence of length 30, there are $20^{30} \approx 10^{39}$ possible sequences. Fortunately, researchers have recently published several databases of AMPs, such as ADP2 [2] and CAMP [3], allowing the creation of large data sets for the computational analysis of AMPs. This analysis can be used to learn patterns in AMP sequences, which we can then use to design novel AMPs. For example, Wang *et al.* [2] used the distribution of amino acids and the most common motifs in the ADP2 database to design a peptide that exhibited strong activity against *E. Coli*. In a linguistically inspired approach, Loose *et al.* [4] used a data set of known AMP sequences to learn a set of regular grammars, and then used those grammars to generate novel peptides.

We propose a method for designing novel AMPs based on learning from *n*-gram counts of classes of amino acid residues, and then using weighted finite-state transducers to produce sequences that include those features that are strongly associated with AMPs. Feature mappings based on *n*-gram counts can be thought of as a representation of a sequence in the frequency domain, and have been used successfully in tasks such as protein remote homology detection [5], the problem of detecting similarities between proteins from different organisms. In our application, we attempt to learn a set of weights that describe how each *n*-gram feature is associated with antimicrobial activity, and then use those weights to generate new sequences. The latter task is made difficult, however, by the fact that most vectors of *n*-gram feature counts do not represent valid peptide sequences. For example, a feature vector that has a positive count for a trigram feature must also have positive counts for the bigrams and unigrams contained within the trigram; otherwise, it cannot represent a valid sequence. If the weight associated with the trigram feature is high, but the weight associated with the bigram features is low or negative, one cannot simply increase the count of the trigram feature while decreasing the count of the bigram features. Weighted finite-state transducers (WFSTs) are a potential solution for this problem in sequence design. Commonly used in speech recognition and natural language processing [6], they can be used to build a weighted lattice of sequences that can be searched or sampled using efficient algorithms to yield valid sequences rich in a desired set of *n*-gram features.

The method we present here is not specific to AMPs and could be applied to many problems in peptide design. We originally developed this framework for a different problem: the creation of peptides capable of binding to inorganic materials such as metals, which is an area with many applications in advanced materials science and nanotechnology [7]. We were able to adapt the framework that we developed for that task to the problem of AMP design with minimal changes. This demonstrates the general applicability of our approach.

II. METHODS

A. Creating a data set

We downloaded all experimentally verified AMPs from the CAMP database as of March 8, 2010, yielding a set of 1,187 peptide sequences. After removing all sequences that contained the nonstandard amino acid letters B, X, and Z, and then extracting representative sequences using the CD-HIT clustering server [8] with a sequence identity parameter of 0.9, we were left with a data set of 862 AMP sequences, with a mean length of 34 amino acid residues and a median length of 30. It is difficult to create a negative training set of experimentally verified non-AMPs, so we followed Thomas *et al.* [3] in noting that AMPs are generally secreted from cells, and downloaded a set of human protein sequences from the UniProt database that were between twenty and fifty amino acids in length, not annotated as antimicrobial, and not annotated as secreted. This gave us a set of 1,224 negative

Christopher Whelan, Brian Roark, and Kemal Sönmez are with the Division of Biomedical Computer Science and Center for Spoken Language Understanding, Oregon Health & Science University, Portland, OR, USA. whelanch@ohsu.edu

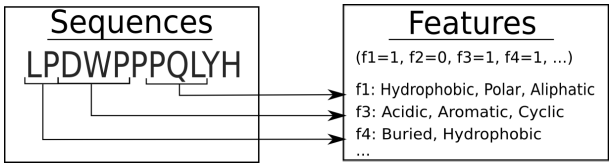


Fig. 1. An example of how we map peptide sequences to features. The substring DWP contributes to the count of the trigram feature $f3$, which represents subsequences of classes "Acidic, Aromatic, Cyclic".

training examples. We randomly split the data, putting 70% in a training set and 30% in a test set.

B. Mapping to a feature space and building a support vector machine classifier

To build a feature space and classifier for AMPs, we define a set of 13 classes to represent the chemical properties of amino acid residues: acidic, cyclic, aliphatic, aromatic, basic, buried, charged, hydrophobic, large, medium, small, non-neutral, and polar. We define unigram, bigram, and trigram features to be the ordered set of classes contained in a subsequence of one, two, or three amino acids. For any sequence, we count the number of times each feature appears, as shown in Figure 1. Given a set of training examples of known AMPs and non-AMPs, we compute vectors of feature counts and train a support vector machine (SVM) using the SVMLight package [9] V6.02. Training an SVM produces a linear classifier in the feature space, defined by

$$w^T \phi(x) + b,$$

where $\phi(x)$ is the mapping of a sequence to the feature space and w is a weight vector that describes the decision boundary hyperplane in the feature space. Given this model, from our trained SVM we extract w, which indicates the direction in the frequency feature space that we hypothesize contain sequences that are more likely to be AMPs.

C. Constructing a WFST to generate novel sequences

A weighted finite-state transducer is an automaton in which each transition between states is associated with an input symbol, an output symbol, and a weight. Formally, they are defined as 8-tuples $(\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$, where Σ is an input alphabet, Δ is an output alphabet, Q is a finite set of states, I is the set of initial states, F is the set of final states, E is the set of transitions between states, λ is a weight function for initial states, and ρ is a weight function for final states. The input and output alphabets are augmented with a symbol ε which represents the empty string, allowing transitions to not input or output a symbol. If no outputs are associated with the transitions then the machine is referred to as a weighted finite-state acceptor (WFSA). Examples of WFSTs and WFSAs can be seen in Figure 2. An important operation on WFSTs is composition, denoted by $\circ$. In composition, the outputs of one WFST are fed to the inputs of a second WFST or WFSA. Efficient implementations of composition allow the construction of complex models based on a set of simple machines [10].

We use WFSTs to generate novel peptide sequences that will score well according to the weight vector learned by our classifier. Our sequence generation system is constructed from three finite-state machines that are composed together. Our first machine, $\mathscr{F}$, is an unweighted transducer that maps from a sequence of amino acids to a sequence of tokens representing features, as shown in Figure 2(a). Our second machine, $\mathscr{S}$, is a WFSA that provides a score for a given sequence of features. This machine is built using the weight vector from the SVM classifier, and accepts each feature with the cost assigned to it by the classifier (Figure 2(b)). Each arc accepts a single feature token f_i with a weight equal to λw_i, where w_i is the value of the classifier weight vector for that feature and λ is a parameter set when building the model. After applying a weight pushing algorithm and normalizing, the weights within the machine are treated as log probabilities; therefore, the parameter λ can be used to vary the "peakedness" of the probability distribution over generated strings because $\lambda \log w_i = \log w_i^{\lambda}$. The third machine, $\mathscr{T}$, is a simple transducer that accepts and outputs any sequence of length 30; this machine constrains the length of our generated sequences. We use a value of 30 because it is the median length of our positive training set.

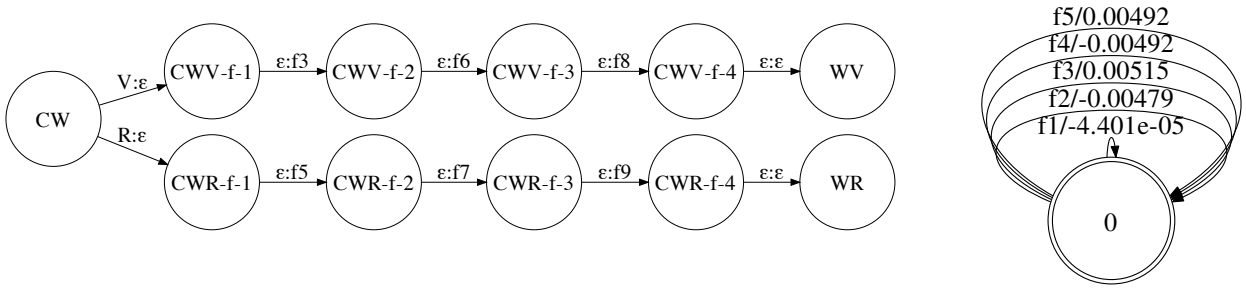
We build our final machine by composing the three sub-machines:

$$\mathscr{T} \circ \mathscr{F} \circ \mathscr{S}$$

This produces a WFST that accepts amino acid sequences of length 30 with a score given by summing up the individual weights of every feature contained within the sequence times λ. We then determinize the paths through this transducer and normalize the scores of each path. We build our finite-state machines using the open source OpenFST library, version 1.1 [11]. In addition to implementing the algorithms needed to produce the transducer as described above, OpenFST provides tools that can search for the highest scoring sequences accepted by the machine, and can sample from high-scoring sequences probabilistically, by treating the scores of each transition within the machine as a negative log probability. Random sampling adds diversity to our results, since the highest scoring sequences generated by the model are often permutations of the same motifs.

D. Generation of novel peptides

We created five sets of novel peptides for testing, each of which consisted of 2,000 novel peptide sequences of length 30. The first, RAND, was a control set, consisting of amino acid residues chosen randomly at each position. As a second control, we also created a set AADIST, which was generated by setting each amino acid independently based on the distribution of amino acids in the CAMP database. We then sampled from our WFST to build three sets WFST1, WFST2, and WFSTNEG, with the λ peakedness parameter set to 1, 2, and -1, respectively. The group WFSTNEG exists to demonstrate the ability of the method to solve the inverse design problem: creating sequences which are unlikely to be



(a) Feature Machine (b) Scorer

Fig. 2. Portions of the finite-state machines that generate new sequences. 2(a): A portion of the finite-state transducer that computes the list of features contained in a sequence, showing paths that can be taken from the state that represents a trigram history of "CW". In one path the machine accepts the amino acid 'V' as input, and emits the features $f3$, $f6$, and $f8$, before proceeding to the state that indicates that the history is now "WV". On the other path the machine accepts the amino acid 'R', and emits the features $f5$, $f7$, and $f9$. The symbol ε represents the empty string. 2(b): A finite-state acceptor that assigns scores to features.

AMPs. For each of our WFST generated groups, we verified that no sequences shared more than 0.9 sequence identity using the CD-HIT clustering web service [8].

E. Evaluating designed AMPs

To evaluate the novel AMP sequences produced by our system, we used the prediction server provided by the CAMP database website [3]. Although not a substitute for experimental validation, using a computational prediction technique allows rapid testing of large sets of peptides, and is therefore useful in validating our approach. The CAMP server classifies peptide sequences as AMPs using three methods: SVMs, random forests (RF), and discriminant analysis (DA). These classifiers were shown to have good performance on a test data set, with overall accuracies of 91.5%, 93.2%, and 87.5%, respectively. The CAMP predictors use a feature set composed of a variety of features including amino acid composition, average hydrophobicity and hydrophilicity of the peptide, transition and composition of groups based on reduced amino acid alphabets, and di- and tri-peptide composition based on hydrophobicity. Therefore, there is partial but not complete overlap with the feature set used in our SVM classifier and WFSTs. Even though this overlap may make it easier for our method to produce sequences that CAMP classifies as AMPs, we believe that the construction of sequences that score highly in a third-party classification system is a valuable demonstration of our approach.

III. RESULTS

A. Performance of the SVM classifier

We began testing our system by evaluating the SVM classifier against our held-out training set of 259 positive and 367 negative examples. Our classifier had an area under the ROC curve of 0.931. This indicates that our feature set of *n*-gram counts of amino acid classes provides sufficient information to train an SVM classifier with strong performance.

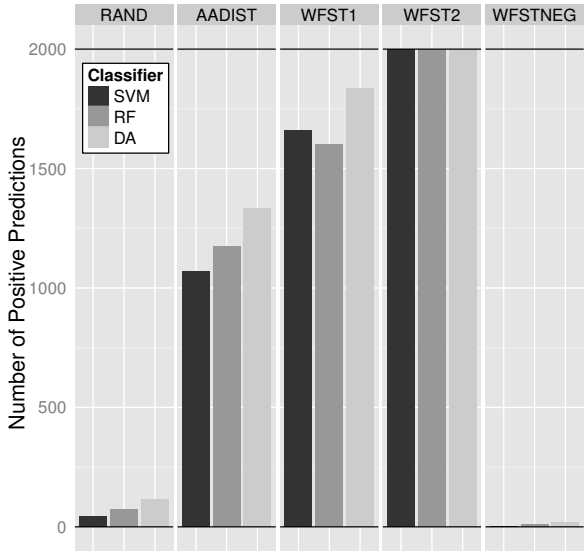


Fig. 3. Number of sequences in a sample of 2000 predicted to be AMPs for different sequence generation methods. A value of 2000 indicates that all of the generated sequences were predicted to be AMPs; a score of 0 indicates that none were predicted to be AMPs. RAND: Sequences generated completely randomly. AADIST: Sequences generated according to the distribution of amino acids in the CAMP database. WFST1: Sequences generated using a WFST with a peakedness parameter λ = 1. WFST2: Sequences generated using a WFST with λ = 2. WFSTNEG: Sequences generated using a WFST with λ = -1. Results are shown for all three prediction methods available at the CAMP database server: SVM: support vector machine. RF: Random Forest. DA: Discriminant Analysis.

B. Number of predicted AMPs in generated sequences

We then determined the number of sequences predicted to be AMPs in our control and test groups using the CAMP prediction servers. The results are shown in Figure 3. Averaged across the three prediction measures, only 3.9% of the random control group were predicted to be AMPs, while 59.6% of the AADIST control set had positive predictions. In the WFST1 group, an average of 84.9% were predicted to be

TABLE I

HIGHEST AND LOWEST SCORING FEATURES IN THE MODEL

Feature	Weight
Cysteine	0.428
Isoleucine	0.279
Lysine	0.246
Aromatic-Buried-Small	0.169
Medium-Medium-Aromatic	0.159
Threonine	-0.323
Leucine	-0.356
Serine	-0.380

AMPs. In the group created with a more peaked probability distribution over sequences, WFST2, an average of 99.9% of the generated sequences were predicted to be AMPs. Finally, in the WFSTNEG group, in which the value of the weights was reversed to reward non-AMP features, only 0.48% of the generated sequences were predicted to be AMPs.

C. Extracting highly weighted features from the model

To better understand the predictions made by our model, we extracted the features with the highest and lowest weights. The five highest and three lowest weighted features are shown in Table I. Unigram features of cysteine, isoleucine, and lysine had heavy positive weights, indicating that those amino acids appear with much greater frequency in the set of AMP sequences than in our negative training set. Also highly weighted were two trigram features: the sequence of an aromatic, a buried, and a small residue, and the sequence of two medium sized residues followed by an aromatic residue. The lowest weighted features were the unigrams of residues that appeared infrequently in AMP sequences.

IV. CONCLUSIONS AND FUTURE WORK

A. Conclusions

We have shown that by using the *n*-gram features of chemical classes of amino acid residues and a trained SVM classifier, we can produce WFSTs that are capable of generating novel sequences which share the same features as the training set. A third-party classification server predicts that a large proportion of these novel sequences will have antimicrobial capabilities. By varying the parameters used to construct our machines, we can exchange diversity of the generated sequences for a higher likelihood of generating new AMPs. We believe that this framework is a promising approach for novel peptide design.

B. Future Work

Although we believe that the use of third-party computational prediction servers shows that this is a promising approach, any attempt to computationally design novel AMPs must eventually be validated by synthesizing and testing actual proteins. We will look to do so in the future.

We believe that this framework is applicable to other problems in peptide design. As mentioned in the introduction, we are also using it to design short peptides that have the ability to bind to inorganic materials. Another potential application is the design of peptides capable of binding to larger proteins, such as G-coupled protein receptors.

We would also like to study the impacts of using different feature mappings for sequences on the performance of the system. Our feature mapping is closely related to string kernel methods such as the spectrum kernel [5] and the mismatch kernel [12], and could easily be extended to incorporate features of other kernels, like the gappy and wildcard [13]. Finally, because our approach relates string kernels and weighted finite-state transducers, it may be possible to improve the construction of our machines using the theory of rational kernels [14], which provides a framework for learning kernels between strings in terms of WFSTs [15].

V. ACKNOWLEDGMENTS

This research was supported in part by NSF Grant #IIS-0811745. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] H. Jenssen, P. Hamill, and R. E. W. Hancock, "Peptide antimicrobial agents," *Clin Microbiol Rev*, vol. 19, pp. 491–511, Jul 2006.

[2] G. Wang, X. Li, and Z. Wang, "APD2: the updated antimicrobial peptide database and its application in peptide design," *Nucleic Acids Research*, vol. 37, no. Database issue, p. D933, 2009.

[3] S. Thomas, S. Karnik, R. S. Barai, V. K. Jayaraman, and S. Idicula-Thomas, "CAMP: a useful resource for research on antimicrobial peptides," *Nucleic Acids Research*, vol. 38, pp. D774–80, Jan 2010.

[4] C. Loose, K. Jensen, I. Rigoutsos, and G. Stephanopoulos, "A linguistic model for the rational design of antimicrobial peptides," *NATURE*, vol. 443, pp. 867–9, Oct 2006.

[5] C. Leslie, E. Eskin, and W. S. Noble, "The spectrum kernel: a string kernel for SVM protein classification," *Pac Symp Biocomput*, pp. 564–75, Jan 2002.

[6] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, Jan 2002.

[7] C. Tamerler and M. Sarikaya, "Molecular biomimetics: nanotechnology and bionanotechnology using genetically engineered peptides," *Philos Transact A Math Phys Eng Sci*, vol. 367, pp. 1705–26, May 2009.

[8] Y. Huang, B. Niu, Y. Gao, L. Fu, and W. Li, "CD-HIT suite: a web server for clustering and comparing biological sequences," *Bioinformatics*, vol. 26, pp. 680–2, Mar 2010.

[9] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning* (B. Schlkopf, C. Burges, and A. Smola, eds.), ch. 11, Cambridge, MA: MIT Press, 1999.

[10] F. C. N. Pereira and M. Riley, *Finite-State Devices for Natural Language Processing*. Cambridge, Massachusetts: MIT Press, 1997.

[11] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, vol. 4783 of *Lecture Notes in Computer Science*, pp. 11–23, Springer, 2007. `http://www.openfst.org`.

[12] C. Leslie, E. Eskin, J. Weston, and W. Noble, "Mismatch string kernels for SVM protein classification," *Advances in Neural Information Processing Systems*, pp. 1441–1448, 2003.

[13] C. Leslie and R. Kuang, "Fast string kernels using inexact matching for protein sequences," *The Journal of Machine Learning Research*, vol. 5, Dec 2004.

[14] C. Cortes, P. Haffner, and M. Mohri, "Rational kernels: Theory and algorithms," *The Journal of Machine Learning*, vol. 5, pp. 1035–1062, Aug 2004.

[15] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning sequence kernels," *Proceedings of the 2008 IEEE Workshop on Machine Learning for Signal Processing, MLSP 2008*, pp. 2–8, Oct 2008.