Identifying Antimicrobial Peptides using Word Embedding with Deep Recurrent Neural Networks

Md-Nafiz Hamid[1,2] and Iddo Friedberg[1,2]

[1]Bioinformatics and Computational Biology Program
[2] Department of Veterinary Microbiology and Preventive Medicine,
Iowa State University, Ames, IA, USA
{nafizh, idoerg}@iastate.edu

January 28, 2018

Abstract

Antibiotic resistance is a major public health crisis, and finding new sources of antimicrobial drugs is crucial to solving it. Bacteriocins, which are bacterially-produced antimicrobial peptide products, are candidates for broadening our pool of antimicrobials. The discovery of new bacteriocins by genomic mining is hampered by their sequences' low complexity and high variance, which frustrates sequence similarity-based searches. Here we use word embeddings of protein sequences to represent bacteriocins, and subsequently apply Recurrent Neural Networks and Support Vector Machines to predict novel bacteriocins from protein sequences without using sequence similarity. We developed a word embedding method that accounts for sequence order, providing a better classification than a simple summation of the same word embeddings. We use the Uniprot/ TrEMBL database to acquire the word embeddings taking advantage of a large volume of unlabeled data. Our method predicts, with a high probability, six yet unknown putative bacteriocins in *Lactobacillus*. Generalized, the representation of sequences with word embeddings preserving sequence order information can be applied to protein classification problems for which sequence homology cannot be used.

1 Introduction

The discovery of antibiotics ranks among the greatest achievements of modern medicine. Antibiotics have eradicated many infectious diseases and enabled many medical procedures that would have otherwise been fatal, including modern surgery, organ transplants, and immunosupressive treatment such as radiation and chemotherapy. However, due to the massive use of antibiotics in healthcare and agriculture, antibiotic resistant bacteria have been emerging in unprecedented scales. Each year, 23,000 people in the US alone die from infections caused by antibiotic resistant bacteria [1]. One strategy to combat antibiotic resistance is to search for antimicrobial compounds other than antibiotics, and which may not be as prone to resistance. A promising class of such compounds are the peptide-based antimicrobials known as bacteriocins[2, 3]. Bacteriocins comprise a broad spectrum of bacterial ribosomal products, and with the increased sequencing of genomes and metagenomes, we are presented with a wealth of data that also include genes encoding bacteriocins. Bacteriocins generally have a narrow killing spectrum making them attractive antimicrobials that would generate less resistance[4].

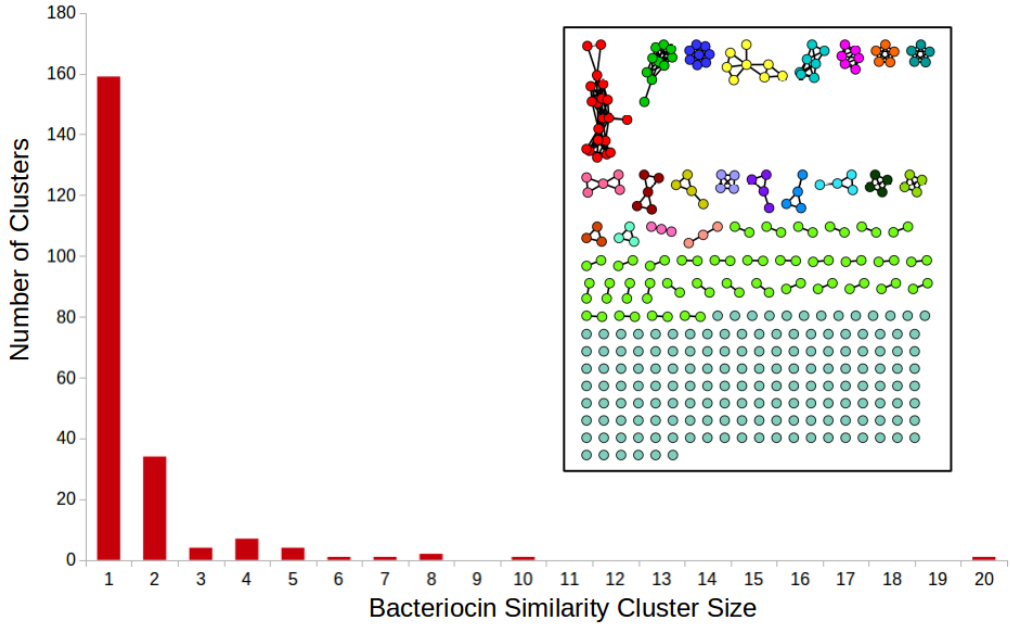


Figure 1: Inset: sequence similarity network for all of the bacteriocins present in the BAGEL dataset. Each node is a bacteriocin. There exists an edge between two nodes if the sequence identity between them is $\geq 35\%$ using pairwise all-*vs*-all BLAST. The barchart shows cluster sizes.

Several computational tools and databases have been developed to aid in the discovery and identification of bacteriocins. BAGEL [5] is a database and a homology-based mining tool that includes a large number of annotated bacteriocin sequences. BACTIBASE [6] is a similar tool, which also contains predicted sequences. AntiSMASH [7] is a platform for genome mining for secondary metabolite producers, which also includes bacteriocin discovery. BOA (Bacteriocin Operon Associator) [8] identifies possible bacteriocins by searching for homologs of *context genes*: genes that are associated with the transport, immunity, regulation, and post-translational modification of bacteriocins. RiPPquest [9] is an automated mass spectrometry based method towards finding Ribosomally synthesized and posttransationally modified peptides (RiPPs) which may include bacteriocins. Recently, MetaRiPPquest [10] was introduced which improves upon RiPPquest by using high-resolution mass spectrometry, and increasing the search space for RiPPs. However, due to their unorthodox structure, bacteriocins are hard to find using standard bioinformatics methods. The challenge in detecting bacteriocins is twofold: (1) a small number of positive examples of known bacteriocin sequences and (2) bacteriocins are highly diverse in sequence, and therefore challenging to discover using standard sequence-similarity based methods, (Figure 1).

To address these challenges we present a novel method to identify bacteriocins using word embedding. We represent protein sequences using Word2vec[11]. After an initial unsupervised learning step, we use supervised learning algorithms to detect novel putative bacteriocins. We use three types of representations

for protein sequences: (1) a simple trigram representation, (2) sum of trigram word-embedding vectors, (3) and the word-embedding vectors as direct inputs into a Recurrent Neural Network (RNN). We show that accounting vector order provides a better performance than other representations, making it a good tool for capturing information in sequences whose similarity is otherwise not detectable.

2 Methods

The method is outlined in Figure 2, and elaborated upon below.

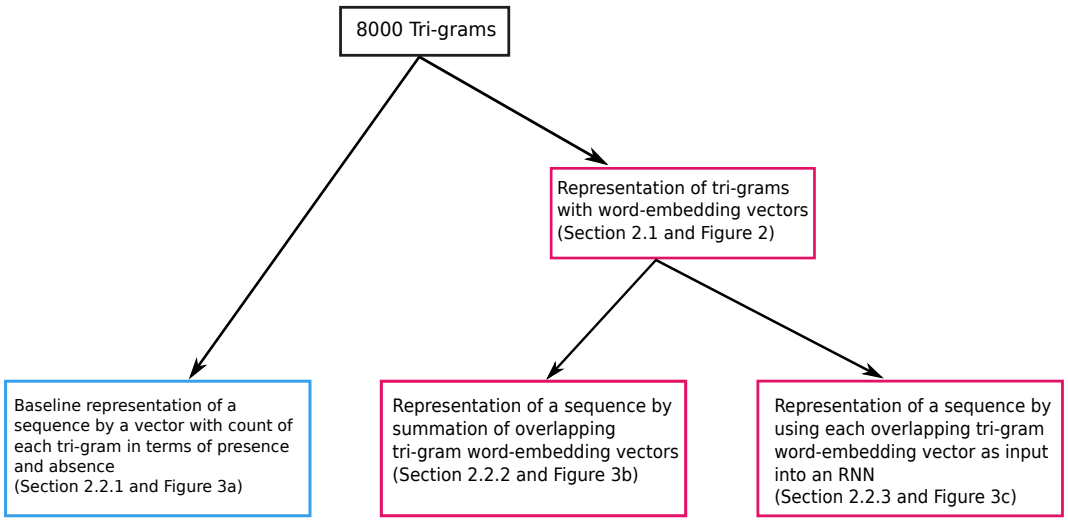


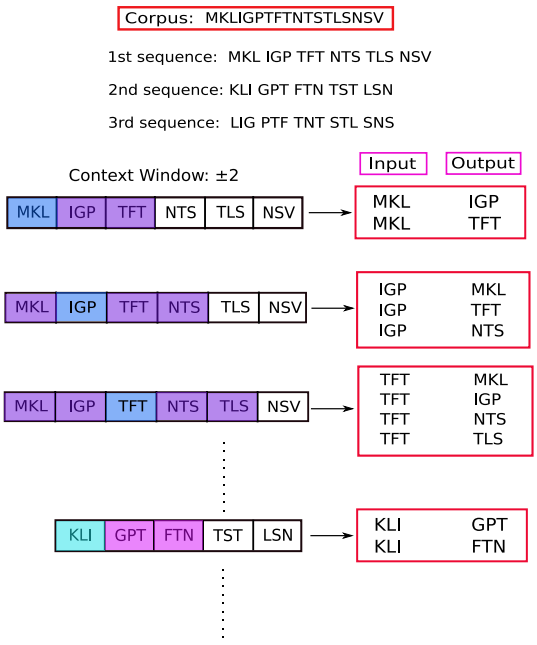
Figure 2: General flowchart of method. See text for details.

2.1 Representation of Proteins with Word Embedding Vectors

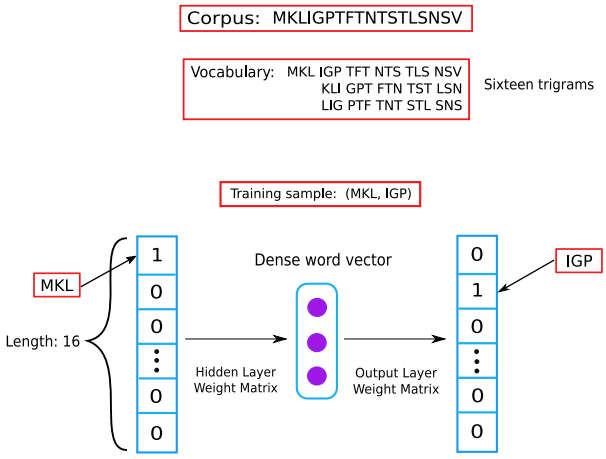
Word embedding is a set of techniques in natural language processing in which words from a vocabulary are mapped onto vectors using a large corpus of text as its input. One word embedding technique is Word2vec, where similar representations are assigned to words that appear in similar contexts based on word proximity as gathered from a large corpus of documents. After training on a large corpus of text, the vectors representing many words show surprising properties. For example, $\overrightarrow{Madrid} - \overrightarrow{Spain} + \overrightarrow{France}$ will result in a vector that is similar to $\overrightarrow{Paris}$, more than other vectors in the corpus [11]. This type of representation have also led to better results in other classification problems, including in biomedical literature classification [12], annotations [13, 14], and genomic sequence classifications [15, 16].

The training for generating the vectors can be done in two ways: the continuous bag of words (CboW) model, or the skip-gram model [11]. We adapted Word2vec for protein representation as in [17], using the skip-gram model. Instead of the common representation of protein sequences as a collection of

counts of n-grams (also known as k-mers) using a 20 letter alphabet, we represent protein sequences using embeddings for each n-gram, covering all possible amino-acid n-grams (we used $n = 3$, leading to $20^3 = 8,000$ trigrams). Each trigram is a "word", and the 8,000 words constitute the vocabulary. The Uniprot/TrEMBL database [18] constitutes the "document corpus".



(a) Generating training instances from an amino acid sequence



(b) Neural network architecture for a single training instance with MKL and input. In practice, all training instances with MKL as input and its context windows neighbors are used to generate the dense word vector associated with MKL.

Figure 3: A simplified example showing representation learning for trigrams with skip-gram training. For simplicity, in the example, the vocabulary comprises of 16 words, the context window is ±2. **(a)** For each sequence in the TrEMBL database we created 3 sequences by starting the sequence from the first, second, and third amino acid as in [17]. This makes sure that we consider all of the overlapping trigrams for a protein sequence. A protein sequence, is then broken into trigrams, and training instances (input, output) are generated according to the size of the context window for the subsequent step of training a neural network. **(b)** The neural network architecture for training on all of the instances generated at (a). The diagram shows training on the instance where MKL is input, and IGP is output which is the first instance generated at (a). At the end of the training, for each trigram a dense word vector of size 200 is produced.

The skip-gram model is a neural network where the inputs and outputs of the network are one-hot vectors with our training instance input word and output word. A one-hot vector is a boolean vector of the size of the vocabulary (8,000 in our case, six in Figure 3b), in which only the entry corresponding to the word of choice is True. We generated the training instances with a context window of size ±5, where we took a word as input and used all of its surrounding words within the context window as outputs. The process is explained in Figure 3. At the end of the training, a 200 dimensional vector for each trigram was generated by the neural network. The goal of this training was to have the 200 dimensional vectors capture information about the surroundings of each trigram that they are representing. In this fashion, we capture the contextual information for each trigram in our corpus of protein sequences. The size

of the vector is a hyper-parameter which we decided upon based on the final supervised classification performance. Vectors of sizes 100, 200, and 300 were generated, and size 200 was chosen. Similarly, context window sizes of 3, 5 and 7 were tested, and size 5 was chosen.

2.2 Representation of sequences

To assess how well we can classify bacteriocins vs. non-bacteriocins, we tested the performance of three different protein sequence representations. For a baseline representation we used a simple trigram representation. To test classification performance, two embedding-based representations: (1) a sum of all dense word vectors of overlapping trigrams in the protein sequence, and (2) an embedding layer with a Recurrent Neural Network.

2.2.1 Baseline Representation - Simple Trigrams

We generated the simple trigram representation to understand the gain of accuracy, if any, using word embedding. The sequence is represented as the count of all overlapping trigrams that comprise it. We created an 8,000 size vector for each sequence where the indices had counts for each occurrence of a trigram in that sequence. Since the vector is sparse, we used truncated Singular Value Decomposition (SVD) to reduce the number of dimensions to 200, and then we applied the classification algorithms to these vectors.

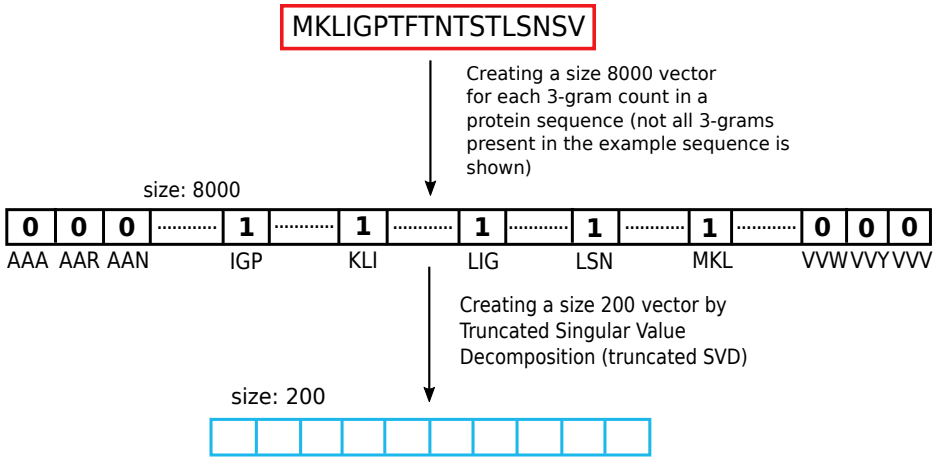


Figure 4: Baseline representation: we represented each protein sequence with the overlapping trigram counts present in that sequence. This leads to a size 8,000 sparse vector. The vector was reduced to a vector of size 200 using Singular Value Decomposition. We used the size 200 vector as the baseline representation.

2.2.2 First Representation with word embedding

Here we represented each protein sequence as the sum of vectors of all overlapping trigrams, a method which was previously shown to be effective for protein family classification [17]. This representation has yielded good results, while limiting the feature vector of a protein sequence to the size of the embedding vector of a single trigram. However, a sum of vectors does not preserve the order of the trigrams. This means that two dissimilar sequences with a similar trigram composition may be represented by similar vectors, which does not capture sequence dissimilarity. Therefore, we also used recurrent neural networks for an ordered way to represent the sequences as detailed below in the second representation.

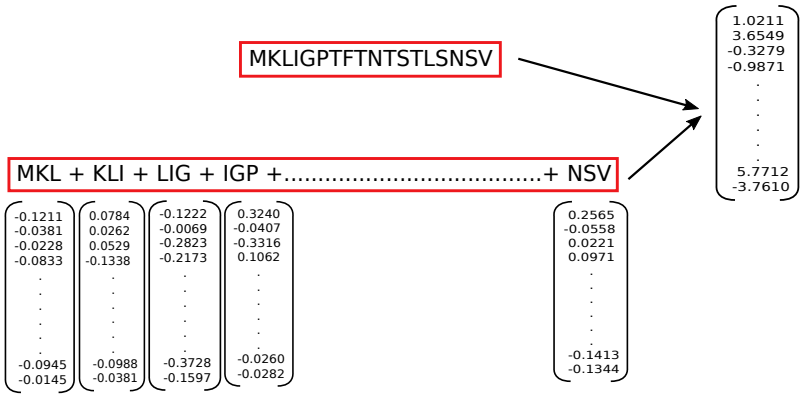


Figure 5: First representation: we used the sum of all overlapping trigram embedding vectors of a protein sequence to represent the sequence itself. This leads to a size 200 embedding vector for each protein sequence. The vectors under each trigram are generated from the training of unlabelled data described at Section 2.1.

2.2.3 Second Representation: An Embedding Layer with a Recurrent Neural Network

Recurrent neural networks (RNN) share the same weights for all inputs in a temporal sequence. We take advantage of this architecture by using a size 200 embedding vector for each overlapping trigram in a protein sequence. Since we are using the embedding vectors of overlapping trigrams as temporal inputs in an RNN, we preserve the order of the trigrams in the protein sequence.

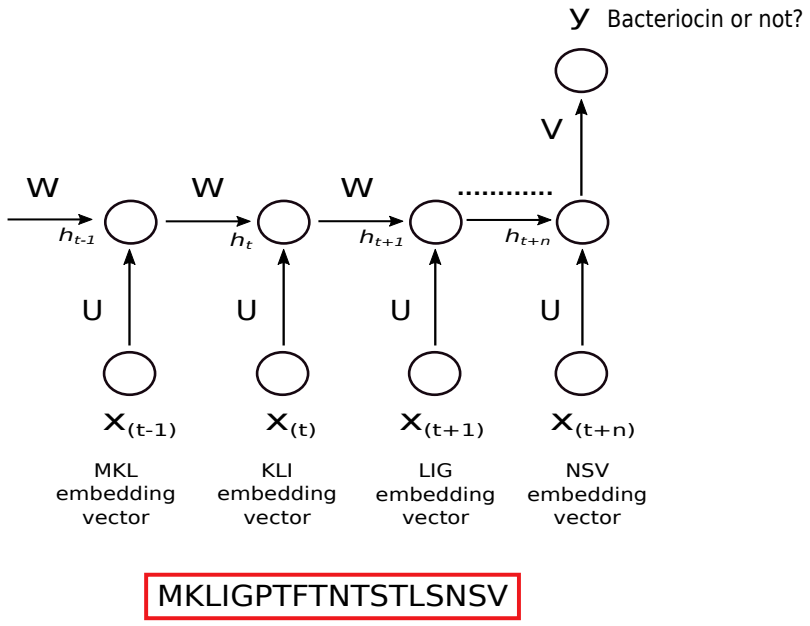
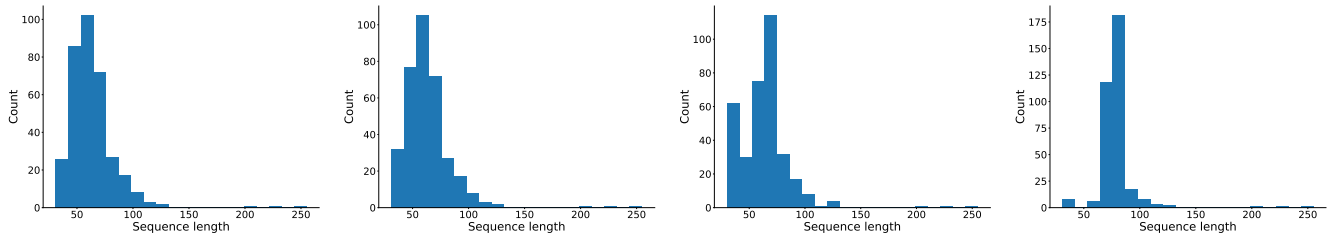


Figure 6: Second representation: we used embedding vectors of each individual overlapping trigram present in a protein sequence as input into a Recurrent Neural Network. $X_{(t)}$ is the input at time step t. In our case, at each time step t, input is the embedding vector of the trigram at that time step. $h_{(t)}$ is the hidden state at time step t. It contains information from the previous inputs as well as the current input. This works like the memory of the network, and because of its mechanism, modern RNNs can preserve information over long ranges unlike traditional models like hidden markov models. U, V, W are weights of the network. As they are being shared over all the inputs, this greatly reduces the number of parameters of the network helping towards generalization. At the end of the sequence, the network produces a prediction y of whether the sequence is a bacteriocin or not. In practice, we used a bidirectional RNN (not shown in figure).

2.3 Building the training dataset

We used 346 sequences of length $\geq$ 30aa from the BAGEL database as our positive bacteriocin training samples. For the negative training set, we used sequences from the Uniprot-Swissprot[19] database. We took all the bacterial protein sequences from this database and used CD-HIT[20] with a 50% identity threshold to reduce redundancy. Then, for the primary negative training set, we took 346 sequences that had the keywords 'not anti-microbial', 'not antibiotic', 'not in plasmid', and that had the same length distribution as our positive bacteriocin sequences. We also generated two additional negative datasets following the same steps as above, with no overlap in the sequences between the three sets. Because identical length sequences were already exhausted by the first negative set, the length distribution of the second and third negative sets are somewhat different than the positive bacteriocin set. Figure 7 shows the length distribution of the positive, and all three negative datasets.



(a) positive set **(b)** primary negative set **(c)** second negative set **(d)** third negative set

Figure 7: Sequence length distributions for the positive bacteriocin set, primary negative set, 2nd, and 3rd negative sets respectively. See text for details.

2.4 Identifying genomic regions for novel putative bacteriocins

To search for genomic regions with a higher probability of containing novel bacteriocins, we took advantage of the biological knowledge of context genes which assist in the transport, modification, and regulation of bacteriocins. Many bacteriocins have some or all of four types of context genes in proximity [21, 22], (Figure 8). Having an experimentally verified set of 54 context genes from [8], we collected the annotation keywords for these 54 context genes from the Refseq database, and BLASTed the BAGEL bacteriocins against the non-redundant protein database. We took the top hits from the result which are essentially the bacteriocins themselves. We then took all the genes with similar keywords to our experimentally verified context gene set surrounding these bacteriocins within a region of ±25kb. After running CD-HIT to remove redundancy, we had 1240 new putative context genes.

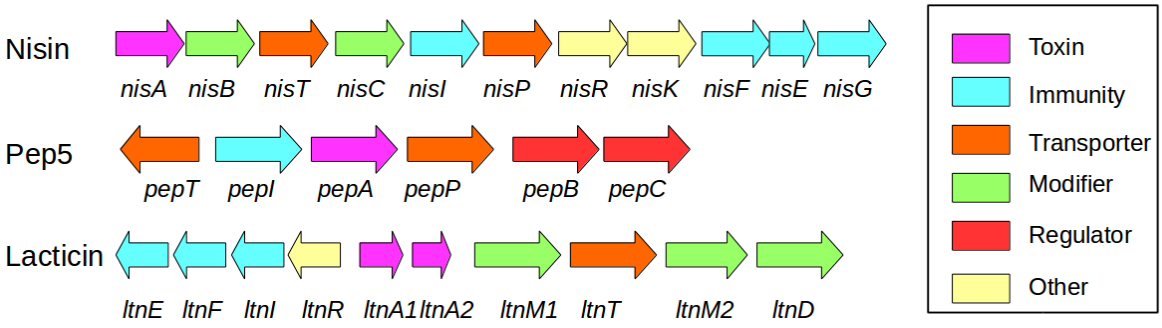


Figure 8: Bacteriocins with context genes. After[21].

We BLASTed all 1294 (54 experimentally verified and 1240 newly found) putative context genes against the whole bacteria RefSeq database [23] and we collected hits with an e-value $\leq 10^{-6}$. We separated all the hits by organism and then arranged them by co-ordinates. Then we identified 50kb regions in the whole genome that have contiguous hits. We applied our trained machine learning model to predict putative bacteriocins in these 50kb regions.

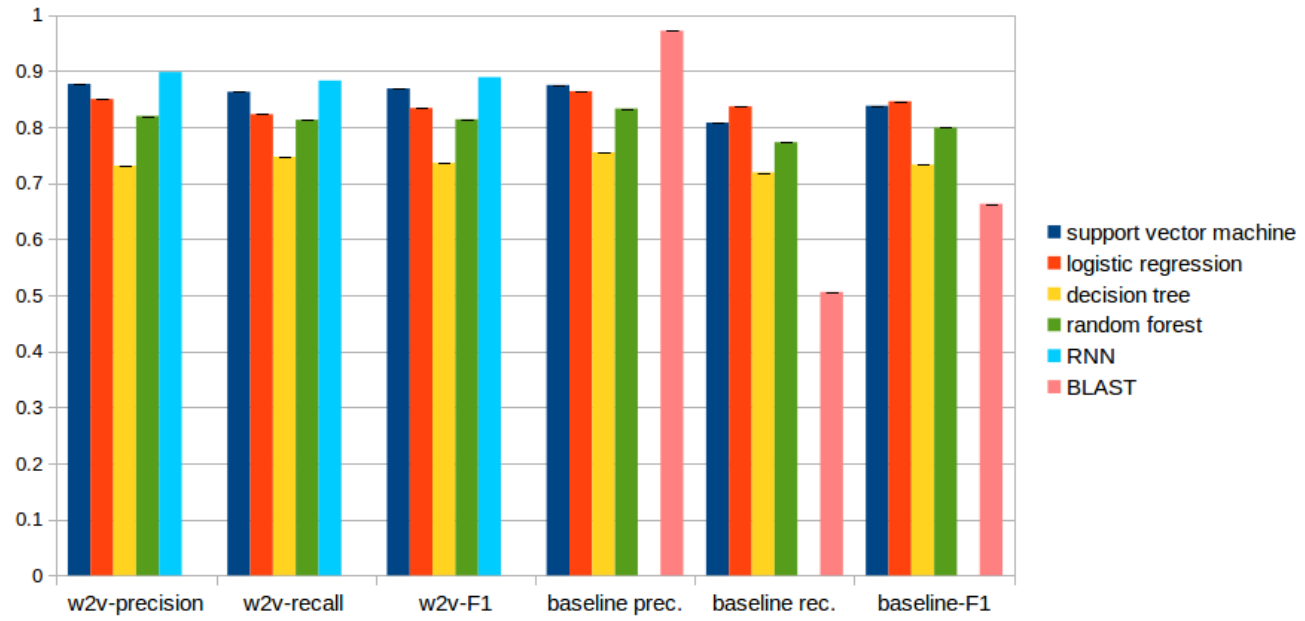


Figure 9: Mean F_1 scores of different algorithms with both Word2vec (w2v) and baseline representation. Error bars are ± standard error. Bidirectional RNN (magenta) provide the best results. See Table S1 for values.

3 Results

3.1 Training dataset

We performed 10× nested cross-validations on the set of positive bacteriocins and the primary negative bacteriocins. We used the representation where each sequence is represented by the summation of its overlapping trigrams for all classification algorithms except the Bidirectional RNN. We also showed BLAST identity score as a baseline. For BLAST, a 35% sequence identity score was used as a threshold. We used the same cross-validation folds for BLAST as other algorithms where we BLASTed the test set against the training set.

For the Bidirectional-RNN, we used the word embeddings of the overlapping trigrams in an ordered manner. The nested cross-validation itself was done 50 times with different random seeds for all cases except for the Bidirectional-RNN for which it was done 10 times due to computational time demand. For BLAST the 10× cross-validation was also done 10 times. The reported results are the mean of 10× nested cross-validation done 50 times (10 times for Bidirec-RNN and BLAST), and the standard error is from those 50 (10 for Bidirec-RNN and BLAST) mean values.

Table S1 and Figure 9 show a comparison of Word2vec and the baseline representations using several classification algorithms. Support vector machine (SVM) with the first Word2vec representation for protein sequences used in section 2.2.1 gives us the best F_1 score while logistic regression works best

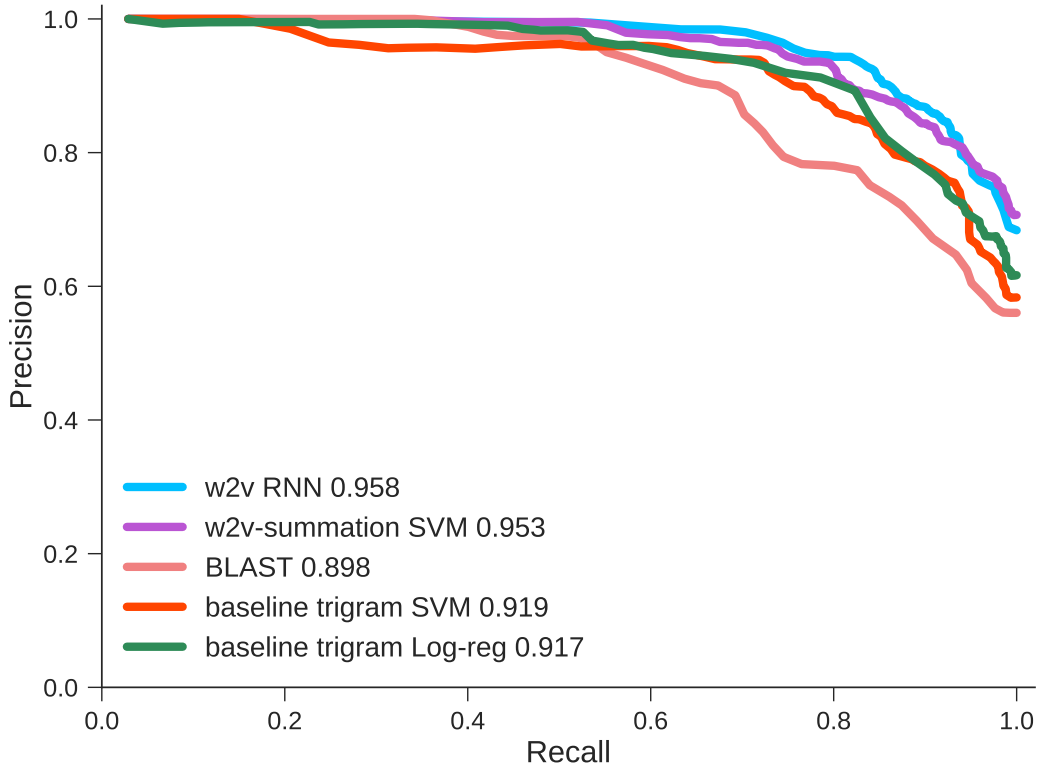


Figure 10: Mean Precision-Recall curves of one run of 10× cross validation for SVM, bidirectional RNN, two trigram baseline models, and BLAST. Number in legend is area under the curve. RNN performs better than unordered Word2vec, and better than the baseline models using simple trigram counts.

for the baseline trigram representation. Word2vec also provides a better balance between precision and recall. All of the methods give better F_1 score than BLAST. Precision (Pr) Recall (Rc) and F_1 are defined as:

$$Pr = \frac{TP}{TP + FP};\ Rc = \frac{TP}{TP + FN};\ F_1 = 2 \times \frac{Pr \times Rc}{Pr + Rc}$$

Where TP: True Positives, FP: False Positives, FN: false negatives. However from Table S1 and Figure10, we can see that Bidirectional Recurrent Neural Networks with an embedded layer representation (second representation) instead of summation (first representation) clearly outperforms all other classification methods. Specifically, we used a 2-layer Bidirectional RNN with Gated Recurrent Units (GRU) to train on our data. Our hyper-parameters of number of neurons, network depth, and dropout [24] rate were determined with nested cross-validation. Since we had a small dataset, we used a dropout rate of 0.5 for the first layer, and 0.7 for the second layer. Both layers had 32 GRU units. We used a fixed number of 100 epochs for training. For optimization, the Adam [25] method was used. After getting the 10× cross-validation results, we trained the RNN on the whole data with the same hyper-parameters. We used the trained RNN to find new bacteriocins in the identified genomic regions of section 2.4.

Table S2 shows the performance comparison of RNN and BLAST for all three negative bacteriocin datasets. The performance gain for RNN over the different sets is large since the second and third negative sets have slightly different length distributions for the negative set than the positive bacteriocin set. In contrast, for BLAST, the performance has gone down over the three different negative datasets. Only for the primary negative set, precision for BLAST is higher which is expected as BLAST predicts only true positives but its recall is low.

Figure 10 shows the Precision-Recall curve for SVM with representation from section 2.2.2, Bidirectional RNN with representation from section 2.2.3. For baselines, SVM and Logistic regression with simple trigram representation, and BLAST with identity score are present. The Word2vec representations have a superior are under curve than the baseline models.

3.2 Results on 50kb Chromosomal Stretches

Since we used two different types of representations for the protein sequences for word2vec, we applied both the SVM (uses sequence representation as summation of the embedding vectors of overlapping trigrams) and RNN (uses sequence representation of separate embedding vectors of each overlapping trigrams in a temporal order) to the identified 50kb regions to predict putative bacteriocins. In both cases, our trained model was trained with the primary negative sequence set. Our motivation for using both representation is, in the first representation we are loosing the order of trigrams, and in the second representation we are preserving the order but we do not know with certainty if preserving the order is beneficial or not. Hence, we use both representations, and if there are predictions that overlap between them, we can be more confident in their potential of being novel bacteriocins.

We plotted the probability distributions for both RNN and SVM as shown in Fig S1. From the violin plot (Figure S1 B), we can see that most of the prediction probabilities lie in the bottom region for both RNN and SVM which is expected.

The RNN model predicted 119 putative bacteriocins with a probability ≥ 0.99. The SVM model predicted 32 putative bacteriocins with a probability of ≥ 0.9. In both cases, BLAST was unable to detect these bacteriocins against the *nr* (non redundant) protein database (e-value $< 10^{-3}$). Together, the RNN and SVM models predicted 145 unique sequences above these thresholds. Between these sequences there was an overlap of 6 sequences. We calculated the Jaccard index for RNN and SVM prediction overlap by-

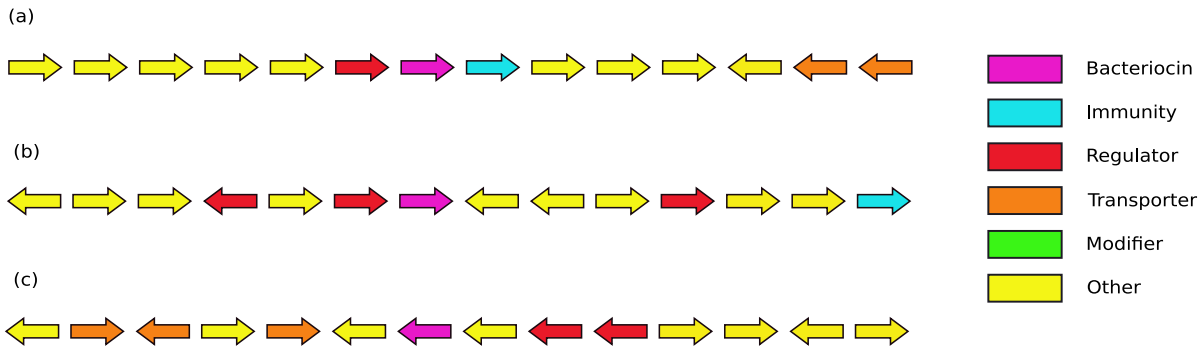


Figure 11: Context genes found surrounding the predicted bacteriocins within ±25kb range. (a) *Lactobacillus acidophilus NCFM* (Locus: NC_006814, putative bacteriocin: YP_193019.1, immunity: YP_193020.1, regulator: YP_193018.1, transporters: YP_193025.1, YP_193026.1) (b) *Lactobacillus helveticus R0052* (GenBnk: NC_018528, putative bacteriocin: YP_006656667.1, immunity: YP_006656674.1, regulator: YP_006656666.1, YP_006656664.1, YP_006656671.1) (c) *Lactobacillus helveticus CNRZ32* (GenBank ID: NC_021744, putative bacteriocin: YP_008236084.1, regulators: YP_008236086.1, YP_008236087.1, transporters: YP_008236082.1, YP_008236080.1, YP_008236079.1)

$$J(RNN > 0.99, SVM > 0.9) = \frac{(RNN > 0.99) \cap (SVM > 0.9)}{(RNN > 0.99) \cup (SVM > 0.9)} = \frac{6}{145} = 0.041$$

Around 4% of the predictions are present in both RNN and SVM predictions. We hypothesize that this low percentage of overlap is due to the presence and absence of order in the two different types of representations used in RNN and SVM respectively. However, we suspect that the 119 sequences predicted by RNN, and 32 sequences predicted by SVM also have a large number potential bacteriocin candidates. Figure 11 shows three of our predicted bacteriocins in their genomic neighborhood *Lactobacillus*. We found several context genes surrounding these predicted bacteriocins, supporting our hypothesis that these bacteriocin predictions are valid.

4 Discussion

We developed a machine learning approach for predicting bacteriocins that does not require sequence similarity searches, and which discovers several putative bacteriocins with a high probability. The different representations take advantage of the large volume of unlabeled bacterial protein sequences available. The trigram representation can be used in other machine learning tasks in computational biology to represent protein sequences. We have also shown performances across two different types of representations of protein sequences with embedding vectors. In the first representation, we used the sum of overlapping trigram embedding vectors to represent a protein sequence. From Table S1 we can see that there is little difference in precision performance between trigram embedding representation and a baseline representation of simple trigram counts. We hypothesize that by summing the embedding vectors we are losing

much of the inherent information present in the biological sequence, which is order-dependent. As a result, our classifiers are not benefiting from the first representation. In contrast, in the second representation, we used the embedding vectors for each overlapping trigram in a protein sequence, and used them as input in a temporal order for a Bidirectional-RNN. This type of embedding vectors together with RNN are used in natural language processing for common tasks such as sentence classification or document classification[26]. Despite the training set being small, with proper regularization our RNN model provides better precision than both baseline and summation of embedding vector representations, and a better recall than the baseline representations. We argue that embedding+RNN can be used to boost the predicting powers of machine learning models in sequence-based classification problems in biology. Our models also provide us with an associated confidence score, which is useful for experimentalists who wish to apply this method towards genome mining of bacteriocins. We chose a threshold of 0.99 for RNN, and 0.9 for SVM to provide the list of putative predictions. Although our training set is balanced in terms of bacteriocins and non-bacteriocins, the number of bacteriocin sequences is much lower. By choosing a high precision threshold experimentalists can avoid unnecessary expenses of synthesizing and functionally testing novel peptides. Finally, we provide six protein sequences that both of our RNN (with probability of $>= 0.99$) and SVM (with probability of $>= 0.9$) model predicted to be putative bacteriocins. We also provide a set of 119 sequences predicted by our RNN with a probability of greater than 0.99, and a set of 32 sequences predicted by our SVM with a probability greater than 0.9. However, due to RNN's superior performance, we recommend any experimentalist who wish to verify the sequences to look at the 119 sequences first. All of these sequences could not be detected against known, and annotated bacteriocins by BLAST when BLASTed against the nr database with an e-value of 10^{-3} or less.

Protein classification tasks are almost always based on sequence similarity and hence, evolutionary relatedness. However, in many cases non-orthologous replacements occur, where two non-homologous proteins perform the same function. Non-orthologous function replacements have been detected using natural language processing [27], genomic context methods[28, 29, 30], and other combined methods[31]. However, such methods require associated metadata or full genomic information. Here we present a solution to find functionally similar non-orthologs that does not require gathering those metadata, but does require a dataset of positive and negative examples. We therefore recommend that word embedding be explored for function classification tasks involving dissimilar biological sequences.

We used the following software tools in this study: Keras [32], Scikit-learn [33], Gensim [34], Matplotlib [35], Jupyter notebooks [36], Numpy and Scipy [37].

Availability

Data and code for this project are available at: `https://github.com/nafizh/Bacteriocin_paper`

Funding

The research is based upon work supported, in part, by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the Army Research Office (ARO) under cooperative Agreement Number W911NF-17-2-0105, and by the National Science Foundation (NSF) grant ABI-1551363. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, ARO, NSF, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Acknowledgements

Will be provided post-review.

References

[1] Centers for Disease Control and Prevention (US). *Antibiotic resistance threats in the United States, 2013*. Centers for Disease Control and Prevention, US Department of Health and Human Services, 2013.

[2] Joanne M. Willey and Wilfred A. van der Donk. Lantibiotics: Peptides of diverse structure and function. *Annual Review of Microbiology*, 61(1):477–501, 2007.

[3] André Guder, Imke Wiedemann, and Hans-Georg Sahl. Posttranslationally modified bacteriocinsthe lantibiotics. *Biopolymers*, 55(1):62–73, January 2000.

[4] Margaret A Riley and John E Wertz. Bacteriocins: evolution, ecology, and application. *Annual Reviews in Microbiology*, 56(1):117–137, 2002.

[5] Auke J van Heel, Anne de Jong, Manuel Montalban-Lopez, Jan Kok, and Oscar P Kuipers. Bagel3: automated identification of genes encoding bacteriocins and (non-) bactericidal posttranslationally modified peptides. *Nucleic acids research*, 41(W1):W448–W453, 2013.

[6] Riadh Hammami, Abdelmajid Zouhir, Christophe Le Lay, Jeannette Ben Hamida, and Ismail Fliss. Bactibase second release: a database and tool platform for bacteriocin characterization. *Bmc Microbiology*, 10(1):22, 2010.

[7] Tilmann Weber, Kai Blin, Srikanth Duddela, Daniel Krug, Hyun Uk Kim, Robert Bruccoleri, Sang Yup Lee, Michael A Fischbach, Rolf Müller, Wolfgang Wohlleben, et al. antismash 3.0a comprehensive resource for the genome mining of biosynthetic gene clusters. *Nucleic acids research*, 43(W1):W237–W243, 2015.

[8] James T Morton, Stefan D Freed, Shaun W Lee, and Iddo Friedberg. A large scale prediction of bacteriocin gene blocks suggests a wide functional spectrum for bacteriocins. *BMC bioinformatics*, 16(1):381, 2015.

[9] Hosein Mohimani, Roland D Kersten, Wei-Ting Liu, Mingxun Wang, Samuel O Purvine, Si Wu, Heather M Brewer, Ljiljana Pasa-Tolic, Nuno Bandeira, Bradley S Moore, et al. Automated genome mining of ribosomal peptide natural products. *ACS chemical biology*, 9(7):1545–1551, 2014.

[10] Hosein Mohimani, Alexey Gurevich, Kelsey L Alexander, C Benjamin Naman, Tiago Leao, Evgenia Glukhov, Nathan A Moss, Tal Luzzatto Knaan, Fernando Vargas, Louis-Felix Nothias, et al. Metarippquest: A peptidogenomics approach for the discovery of ribosomally synthesized and post-translationally modified peptides. *bioRxiv*, page 227504, 2017.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[12] José Antonio A. Minarro-Giménez, Oscar Marín-Alonso, and Matthias Samwald. Exploring the application of deep learning techniques on medical text corpora. *Studies in health technology and informatics*, 205:584–588, 2014.

[13] Magdalena Zwierzyna and John P. Overington. Classification and analysis of a large collection of in vivo bioassay descriptions. *PLoS computational biology*, 13(7), July 2017.

[14] Dat Duong, Eleazar Eskin, and Jessica Li. A novel word2vec based tool to estimate semantic similarity of genes by using gene ontology terms. *bioRxiv*, 2017.

[15] Maria Katherine Mejia Guerra and Edward S. Buckler. k-mer grammar uncovers maize regulatory architecture. *bioRxiv*, 2017.

[16] Aparajita Dutta, Tushar Dubey, Kusum Kumari Singh, and Ashish Anand. Splicevec: distributed feature representations for splice junction prediction. *bioRxiv*, 2017.

[17] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287, 2015.

[18] Rolf Apweiler, Amos Bairoch, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl 1):D115–D119, 2004.

[19] Emmanuel Boutet, Damien Lieberherr, Michael Tognolli, Michel Schneider, and Amos Bairoch. Uniprotkb/swiss-prot: the manually annotated section of the uniprot knowledgebase. *Plant bioinformatics: methods and protocols*, pages 89–112, 2007.

[20] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, and Weizhong Li. Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

[21] W. M. de Vos, O. P. Kuipers, J. R. van der Meer, and R. J. Siezen. Maturation pathway of nisin and other lantibiotics: post-translationally modified antimicrobial peptides exported by gram-positive bacteria. *Molecular microbiology*, 17(3):427–437, August 1995.

[22] O. McAuliffe, R. P. Ross, and C. Hill. Lantibiotics: structure, biosynthesis and mode of action. *FEMS microbiology reviews*, 25(3):285–308, May 2001.

[23] Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research*, 35(suppl_1):D61–D65, 2006.

[24] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Rie Johnson and Tong Zhang. Supervised and semi-supervised text categorization using lstm for region embeddings. In *International Conference on Machine Learning*, pages 526–534, 2016.

[27] Karin M. Verspoor, Judith D. Cohn, Komandur E. Ravikumar, and Michael E. Wall. Text mining improves prediction of protein functional sites. *PloS one*, 7(2):e32171+, February 2012.

[28] R. Overbeek, M. Fonstein, M. D'Souza, G. D. Pusch, and N. Maltsev. The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences of the United States of America*, 96(6):2896–2901, March 1999.

[29] M. Huynen, B. Snel, W. Lathe, and P. Bork. Predicting protein function by genomic context: quantitative evaluation and qualitative inferences. *Genome research*, 10(8):1204–1210, August 2000.

[30] François Enault, Karsten Suhre, and Jean-Michel M. Claverie. Phydbac "gene function predictor": a gene annotation tool based on genomic context analysis. *BMC bioinformatics*, 6(1):247+, October 2005.

[31] Andrea Franceschini, Damian Szklarczyk, Sune Frankild, Michael Kuhn, Milan Simonovic, Alexander Roth, Jianyi Lin, Pablo Minguez, Peer Bork, Christian von Mering, and Lars J. Jensen. String v9.1:

protein-protein interaction networks, with increased coverage and integration. *Nucleic acids research*, 41(Database issue):D808–D815, January 2013.

[32] François Chollet et al. Keras, 2015.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

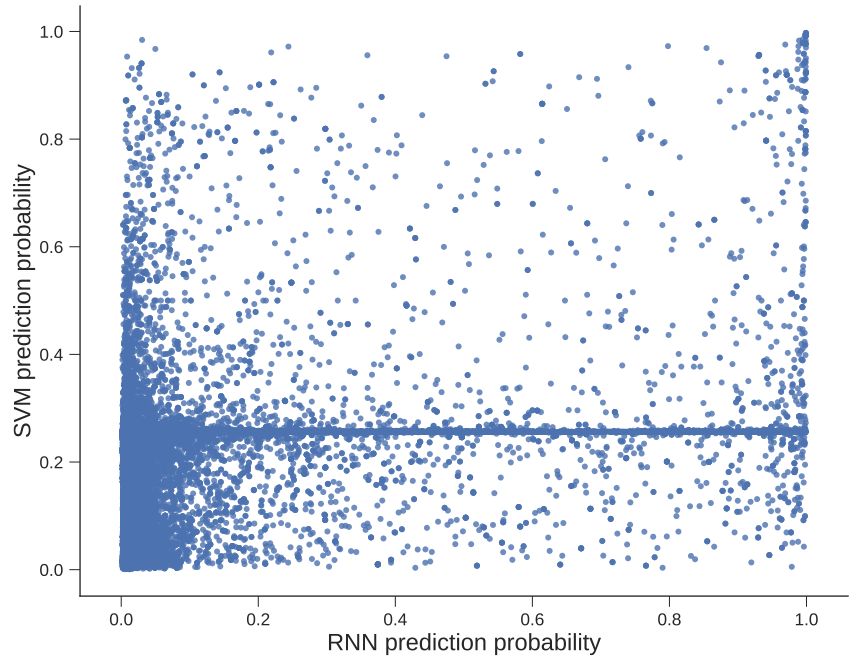
[34] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[35] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

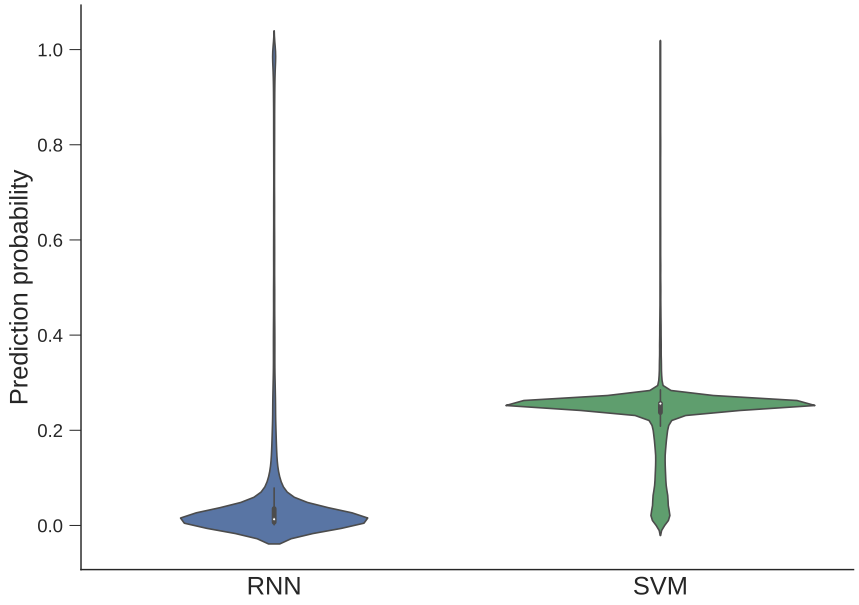
[36] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.

[37] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

Supplementary Material



(a)



(b)

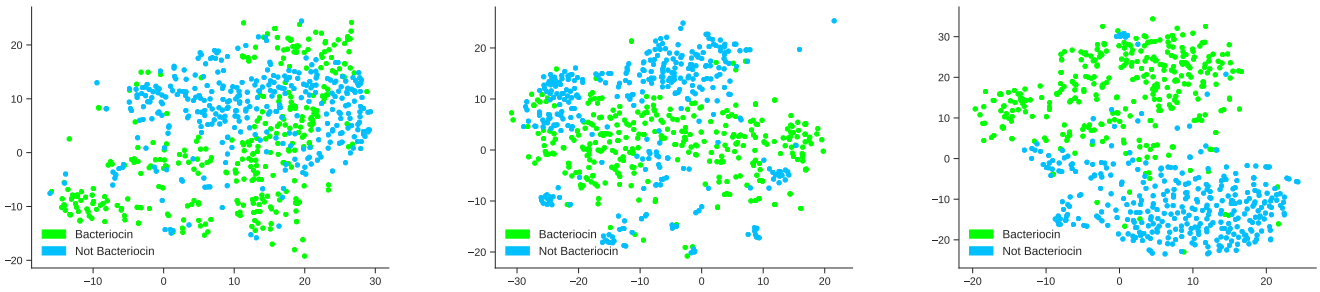
Figure S1: (a) Scatter plot between the RNN and SVM prediction probabilities for sequences from all 50kb regions. **(b)** Violin plots showing the distribution of the prediction probabilities for both RNN and SVM.

Table S1: Comparison between Word2vec and baseline representation. SVM:Support Vector Machine; LogReg: logistic regression; DT: decision tree; RF: random forest, RNN: Bidirectional Recurrent Neural Network. The baseline uses the protein sequence representation described at section 2.2.1. The Bidirectional RNN uses the ordered representation for protein sequences mentioned in section 2.2.3. For the word2vec representation, all other classification algorithms use the protein sequence representation described at section 2.2.2

	word2vec			**baseline - trigram count**		
	Mean Precision	**Mean Recall**	**Mean F1**	**Mean Precision**	**Mean Recall**	**Mean F1**
SVM	0.877 ± 0.001	0.863 ± 0.0011	0.869 ± 0.0009	0.875 ± 0.001	0.808 ± 0.002	0.838 ± 0.001
LogReg	0.85 ± 0.001	0.823 ± 0.001	0.834 ± 0.0009	0.864 ± 0.002	0.837 ± 0.002	0.846 ± 0.001
DT	0.731 ± 0.002	0.747 ± 0.003	0.736 ± 0.002	0.755 ± 0.015	0.719 ± 0.002	0.733 ± 0.001
RF	0.82 ± 0.001	0.813 ± 0.002	0.814 ± 0.001	0.833 ± 0.002	0.773 ± 0.001	0.799 ± 0.001
RNN	**0.898 ± 0.003**	**0.883 ± 0.003**	**0.889 ± 0.001**	NA	NA	NA

Table S2: Comparison of RNN and BLAST performance with three different negative sets

	BLAST			**RNN**		
	Mean Precision	Mean Recall	Mean F1	Mean Precision	Mean Recall	Mean F1
Primary negative dataset	**0.972 ± 0.0006**	0.506 ± 0.002	0.663 ± 0.002	0.898 ± 0.003	**0.883 ± 0.003**	**0.889 ± 0.001**
Second negative dataset	0.909 ± 0.002	0.504 ± 0.002	0.645 ± 0.001	**0.924 ± 0.002**	**0.898 ± 0.001**	**0.909 ± 0.001**
Third negative dataset	0.747 ± 0.004	0.504 ± 0.002	0.599 ± 0.002	**0.937 ± 0.002**	**0.921 ± 0.002**	**0.928 ± 0.002**



(a) **(b)** **(c)**

Figure S2: t-sne visualization of word vector representations for positive and negative bacteriocin amino acid sequences with **a-c** as first, second and third negative data sets. This is the representation where protein sequences are represented by the summation of their overlapping 3-grams(Section 2.2.2).