Bioinformatics, 35(12), 2019, 2009–2016
doi: 10.1093/bioinformatics/bty937
Advance Access Publication Date: 10 November 2018
Original Paper

Genome analysis

Identifying antimicrobial peptides using word embedding with deep recurrent neural networks

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

[1]Interdepartmental program in Bioinformatics and Computational Biology and [2]Department of Veterinary Microbiology and Preventive Medicine, Iowa State University, Ames, IA 50011, USA

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on June 18, 2018; revised on August 27, 2018; editorial decision on November 2, 2018; accepted on November 8, 2018

Abstract

Motivation: Antibiotic resistance constitutes 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 the available choices of antimicrobials. However, the discovery of new bacteriocins by genomic mining is hampered by their sequences' low complexity and high variance, which frustrates sequence similarity-based searches.

Results: Here we use word embeddings of protein sequences to represent bacteriocins, and apply a word embedding method that accounts for amino acid order in protein sequences, to predict novel bacteriocins from protein sequences without using sequence similarity. 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 peptide and protein classification problems for which sequence similarity cannot be used.

Availability and implementation: Data and source code for this project are freely available at: https://github.com/nafizh/NeuBI.

Contact: idoerg@iastate.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

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 treatments. However, due to the prevalent 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 (Centers for Disease Control and Prevention, US Department of Health and Human Services, 2013). 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 (Guder *et al.*, 2000; Willey and van der Donk, 2007). 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 (Riley and Wertz, 2002).

Several computational tools and databases have been developed to aid discovery and identification of bacteriocins. BAGEL (van Heel *et al.*, 2013) is a database and a homology-based search tool that includes a large number of experimentally verified annotated bacteriocin sequences. BACTIBASE (Hammami *et al.*, 2010) is a similar tool, which also contains predicted sequences. AntiSMASH (Weber *et al.*, 2015) is a platform for genome mining for secondary

metabolite producers, which also includes bacteriocin discovery. BOA (Bacteriocin Operon Associator) (Morton *et al.*, 2015) 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 (Mohimani *et al.*, 2014) is an automated mass spectrometry based method towards finding Ribosomally synthesized and posttransationally modified peptides (RiPPs) which may include bacteriocins. Recently, MetaRiPPquest (Mohimani *et al.*, 2017) improved upon RiPPquest by using high-resolution mass spectrometry, and increasing the search space for RiPPs. However, bacteriocins are still hard to identify using standard bioinformatics methods. The challenge in detecting bacteriocins is twofold: first, a small number of positive examples of known bacteriocin sequences, and second, bacteriocins are highly diverse in sequence, and therefore challenging to discover using standard sequence-similarity based methods (Fig. 1).

To address these challenges we present a novel method to identify bacteriocins using word embedding. We represent protein sequences using Word2vec (Mikolov *et al.*, 2013). Using this representation, we use a deep Recurrent Neural Network (RNN) to distinguish between bacteriocin and non-bacteriocin sequences. Our results show that a word embedding representation with RNNs can classify bacteriocins better than current tools and algorithms for biological sequence classification.

2 Materials and methods

2.1 The 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 represented as vectors using a large corpus of text as the input. One word embedding technique is Word2vec, where similar vector 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 interesting and useful contextual properties. For example, after training on a large corpus of English language documents, given vectors representing words that are countries and capitals, $\overrightarrow{Madrid} - \overrightarrow{Spain} + \overrightarrow{France}$ will result in a vector that is similar to $\overrightarrow{Paris}$, more than other vectors in the corpus (Mikolov *et al.*, 2013). This type of representation has led to better performance in downstream classification problems, including in biomedical literature classification (Chen *et al.*, 2018; Minarro-Giménez *et al.*, 2014), annotations (Duong *et al.*, 2018; Zwierzyna and Overington, 2017) and genomic sequence classifications (Dutta *et al.*, 2018; Du *et al.*, 2018; Mejia Guerra and Buckler, 2017; Zhang *et al.*, 2018).

The training for generating the vectors can be done in two ways: the continuous bag of words (CboW) model, or the skip-gram model (Mikolov *et al.*, 2013). We adapted Word2vec for protein representation as in (Asgari and Mofrad, 2015), 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 $20^3 = 8000$ trigrams). Each trigram is a 'word', and the 8000 words constitute the vocabulary. The Uniprot/TrEMBL database (Apweiler *et al.*, 2004) constitutes the equivalent of the document corpus.

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 (8000 in our case, six in Fig. 2), in which only the entry corresponding to the word of choice has a value of True. We generated the training instances using a context window of size ± 5, where we took a word as input (in this case, a word is a trigram), and used the surrounding words within the context window as outputs. The process is explained in Figure 2. 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

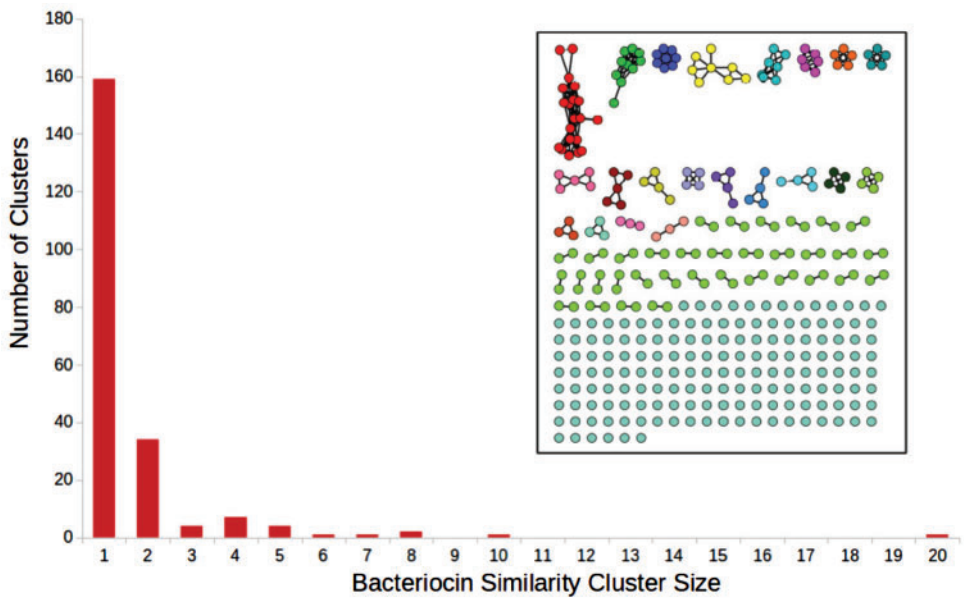

Fig. 1. Inset: sequence similarity network for all of the bacteriocins present in the BAGEL dataset. Each node is a bacteriocin. There exists an edge between two nodes if the sequence identity between them is $\geq$ 35% using pairwise all-versus-all BLAST. The bar chart shows cluster sizes

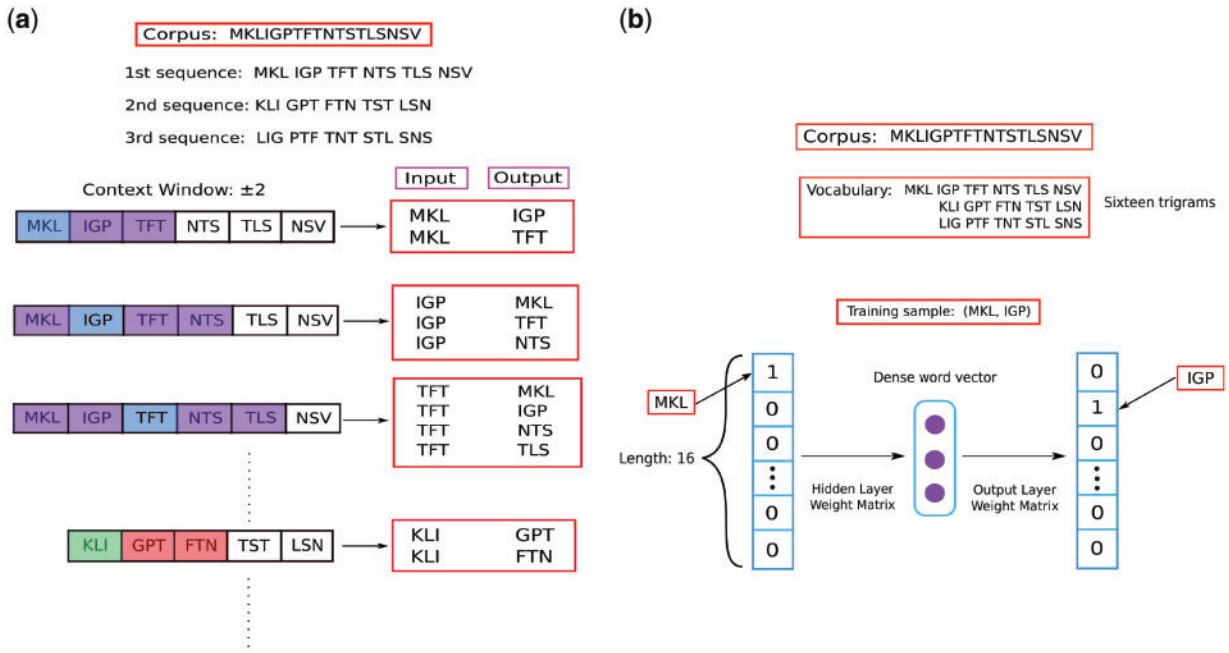

Fig. 2. A simplified example showing representation learning for trigrams with skip-gram training. For simplicity, in the example, the vocabulary comprises of 16 words, the context window is ±2 (in our study, the vocabulary size was 8000 and the context window ±5). (**a**) For each sequence in the TrEMBL database we created 3 sequences by starting the sequence from the first, second and third amino acid as in (Asgari and Mofrad, 2015). 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 (center, purple circles). (Color version of this figure is available at *Bioinformatics* online.)

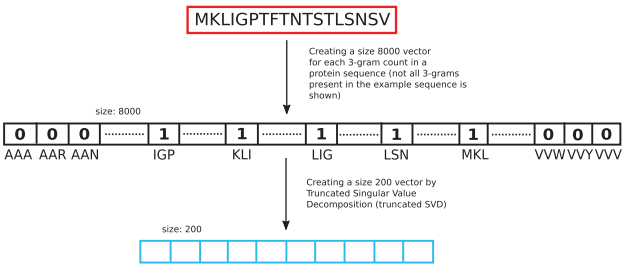

Fig. 3. We represented each protein sequence with the overlapping trigram counts present in that sequence. This leads to a size 8000 sparse vector. The vector was reduced to a vector of size 200 using Singular Value Decomposition. We used the size 200 vector as the baseline representation

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 hyperparameter 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 (Fig. 3). Similarly, context window sizes of 3, 5 and 7 were tested, and size 5 was chosen.

2.2 Word2vec with a recurrent neural network

Taking the word-embedding representation for each trigram present in a protein sequence, we used a Recurrent Neural Network (RNN) to take all trigram embedding vectors as its input to represent a protein sequence (Fig. 4). Since RNNs share the same weights for all inputs in a temporal sequence, we took advantage of this architecture by using an embedding vector of size 200 for

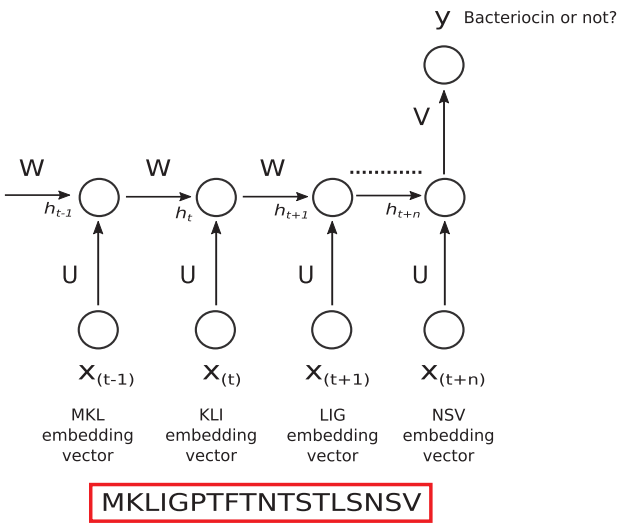

Fig. 4. We used embedding vectors of each individual overlapping trigram present in a protein sequence as input into a Recurrent Neural Network. $X_{(t)}$ is the input at time step *t*. In our case, at each time step *t*, input is the embedding vector of the trigram at that time step. $h_{(t)}$ is the 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)

each overlapping trigram in a protein sequence. By using the embedding vectors of overlapping trigrams as temporal inputs to an RNN, we preserved the order of the trigrams in the protein sequence. Regarding the architecture of the RNN, we used a two-layer Bidirectional RNN with Gated Recurrent Units (GRU) to train on our data. Our hyper-parameters of number of neurons, network depth and dropout (Srivastava *et al.*, 2014) 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 which was also decided by nested cross-validation. For optimization, the Adam (Kingma and Ba, 2014) method was used.

2.3 Comparing with baseline methods

We compared the performance of our method with four baseline methods: (i) a simple trigram representation, (ii) an averaged word-embedding representation, (iii) BLAST (Altschul *et al.*, 1997) and (iv) HMMER3 (Eddy, 2011).

We used a trigram representation of sequences in bioinformatics to understand the gain of accuracy, if any, of using word embedding over simple trigram based representation. To implement the simple trigram representation, we created an 8000 size vector for each sequence where the indices had counts for each occurrence of a trigram in that sequence. In this representation, the order of the trigrams is not preserved. Since the vector is sparse, we used truncated Singular Value Decomposition (SVD) to acquire the most importance features, and reduce the size of the vector. We tried vector sizes of 100 and 200, and used 200 as it led to better classification performance. We then used these vectors with a support vector machine (SVM), logistic regression (LR), decision tree (DT) and random forest (RF), to classify genes into bacteriocins and non-bacteriocins.

We also used an averaged word-embedding representation, and evaluated its performance with SVM, LR, DT and RF. We summed the embedding vectors for each overlapping trigram in a sequence, and divided the sum by the length of the sequence. We then used this new mean embedding vector that is representative of the whole protein sequence with supervised learning algorithms.

We compared the performance of our method with BLAST, the method of choice for sequence similarity search and, by proxy, determination of gene and protein function. We use BLAST to see if machine learning based, alignment free methods do indeed improve performance over alignment based methods to identify potential bacteriocins. We used a 35% sequence identity score as a threshold to assign a bacteriocin label to a protein sequence. This threshold was used to increase BLAST's recall even at the expense of precision, and was based on the finding that 35% ID is the 'Twilight Zone' of protein sequence alignments below which one cannot unambiguously distinguish between true and false sequence alignment, using protein structure as a standard (Rost, 1999).

We also compared our performance with another popular alignment based method, HMMER3, which constructs profile hidden Markov models or pHMMs from multiple sequence alignments. In turn, the pHMMs serve as an accurate tool for sequence searching. Here we used bacteriocin pHMMs which we constructed using BOA (Morton *et al.*, 2015). BOA uses the BAGEL (van Heel *et al.*, 2013) dataset, and its homologs (BLAST e-value $< 10^{-5}$) against GenBank (Benson *et al.*, 2014) bacterial database to build bacteriocin-specific pHMMs. We used the HMMSearch functionality provided by HMMER3, and use the pHMMs from BOA to measure performance against our test set in terms of precision, recall and F_1 score.

2.4 Building the training dataset

We used 346 experimentally determined bacteriocin sequences of lengths $\geq$ 30aa from the BAGEL database as our positive bacteriocin training samples. For the negative training set, we used sequences from the Uniprot-Swissprot (Boutet *et al.*, 2016) database. We took all the bacterial protein sequences from this database and used CD-HIT (Fu *et al.*, 2012) 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 5 shows the length distribution of the positive, and all three negative datasets.

2.5 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 known proximity of *context genes* whose products assist in the transport, modification and regulation of bacteriocins. Many bacteriocins have some or all of four types of context genes in proximity (de Vos *et al.*, 1995; McAuliffe *et al.*, 2001) (Fig. 6). Having an experimentally verified set of fifty-four context genes from (Morton *et al.*, 2015), we now aimed to expand it. To do so, we collected the annotation keywords for these context genes from the Refseq database,

We ran BLAST using all 1294 (54 experimentally verified and 1240 newly found) putative context genes against the whole bacteria RefSeq database (Pruitt *et al.*, 2006) and we collected hits with an e-value $\leq 10^{-6}$. We separated all the hits by organism, arranged them by co-ordinates and identified 50 kb regions in the whole

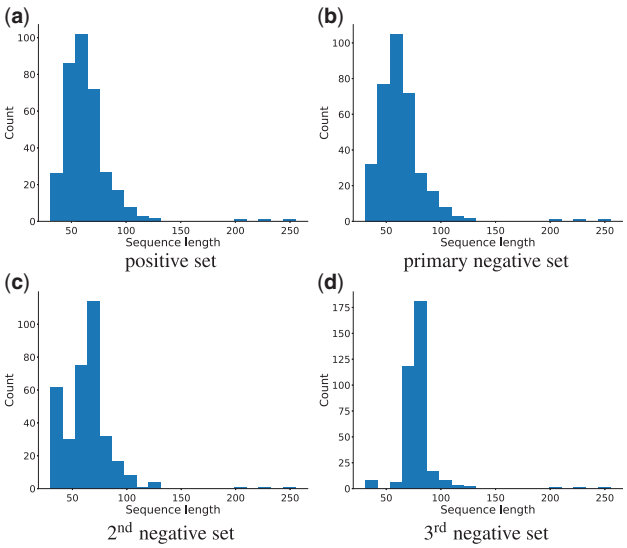

Fig. 5. Sequence length distributions for the positive bacteriocin set, primary negative set, 2nd and 3rd negative sets respectively. Mean lengths of training sets were 63.57aa (with the primary negative set), 63.53 (with second negative set) and 70.92 (with third negative set). See text for details

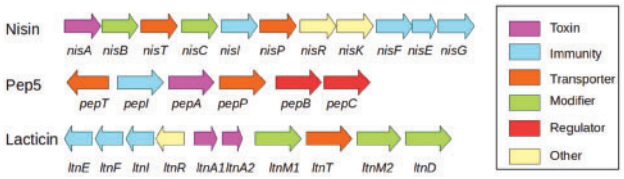

Fig. 6. Bacteriocins with context genes. After de Vos *et al.* (1995)

genome that have contiguous hits. We then ran our method on the ORFs that were not identified as conext genes, to see if we could identify new bacteriocin genes within these regions.

We used the following software tools in this study: Keras (Chollet *et al.*, 2015), Scikit-learn (Pedregosa *et al.*, 2011), Gensim (Řehůřek and Sojka, 2010), Matplotlib (Hunter, 2007), Jupyter notebooks (Kluyver *et al.*, 2016), Biopython (Cock *et al.*, 2009), Numpy and Scipy (Walt *et al.*, 2011).

We then took all the genes with similar keywords to our experimentally verified context gene set surrounding the BAGEL bacteriocins within a region of ±25kb. After running CD-HIT (Li and Godzik, 2006) to remove redundancy, we had 1240 new putative context genes.

2.6 Datasets

We performed 10× cross-validations on the three datasets we built where the datasets consist of positive bacteriocins from BAGEL, and the three negative datasets we built from Uniprot Swissprot database.

The cross-validation itself was done 50 times with different random seeds for all cases except for the RNN, BLAST and HMMER for which it was done 10 times due to computational time demand. For BLAST, a 35% sequence identity score was used as a threshold for calling a result positive. We used the same cross-validation folds for BLAST as other algorithms where we BLASTed the test set against the training set. For HMMER, an e-value of $< 10^{-3}$ was used as the threshold for deciding if a sequence is bacteriocin. The reported results are the mean of 10× nested cross-validation done 50 times (10 times for RNN, BLAST and HMMER), and the standard error is from those 50 (10 for RNN, BLAST and HMMER) mean values.

3 Results

Supplementary Table S1 and Figure 7 show a comparison of Word2vec, trigram representation, BLAST and HMMER for predicting bacteriocins using the primary bacteriocin dataset in terms of precision, recall and F_1 score.

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.

W2v+RNN provides the best recall, and F_1 score. HMMER and BLAST have better precision scores, which is expected as they only predict true positives by e-value and sequence identity respectively but they have high false negative rate. Using a simple trigram representation, SVM and LR perform similarly but with lower precision, recall and F_1 score than w2v+RNN. Using mean Word2vec representation as input, LR, DT and RF provides similar or worse F_1 score than the performances of those supervised methods with a simple trigram representation. Only mean Word2vec representation as input to an SVM shows a competitive performance against W2v+RNN. Still, the difference in F_1 between mean w2v+SVM and Word2vec+RNN was statistically significant and shows that Word2vec+RNN performs better (one sided t-test, $P = 5.48 \times 10^{-8}$).

Figure 8 shows the precision-recall curves for w2v+RNN, averaged Word2vec representation with SVM, LR, RF. Also, simple trigram representation with SVM (trigram+SVM), trigram+LR, trigram+RF and BLAST. RNN has the largest area under the curve. W2v+SVM is competitive with w2v+RNN as was also seen in Figure 7. The curve for HMMER could not be shown as we need a confidence value for each prediction which HMMER does not provide.

Supplementary Tables S2 and S3 show the performance differences using the two other training datasets. The length distribution of the protein sequences in the positive and negative sets are different as mentioned in Section 2. Looking at Supplementary Table S2, the improvement in the w2v+RNN and the trigram based methods is evident, as well as the precision of HMMER. We assume the length disparity between the positive and negative sequences have helped in correctly classifying bacteriocins. Surprisingly, the precision of BLAST has decreased compared with its precision in the primary bacteriocin dataset. The performance of HMMER has largely remained the same over the different negative sets, with its predictions remaining more or less the same because of its low false positive rate. Supplementary Table S3 shows the performance comparison for the third bacteriocin dataset. The length disparity between positive and negative sequences for the third dataset is even greater than the second bacteriocin dataset. SVM, LR, DT and RF have all improved performance. SVM's precision is comparable to that of w2v+RNN. RNN still has the best recall and F_1 score. In contrast, BLAST's performance has significantly decreased indicating that somehow the length disparity is causing problems in identifying true bacteriocins. Just like the second bacteriocin dataset, HMMER's performance remains almost the same with a slight improvement on the precision score.

After evaluating all the methods, we trained the best performing method, w2v+RNN on the whole dataset with the same hyperparameters, and this final trained RNN was used to find new bacteriocins in the 50 kb genomic regions that are that were identified based on context genes and are suspected of containing bacteriocin genes.

3.1 Results on 50 kb chromosomal stretches

We applied our trained w2v+RNN model on the sequences identified from the 50 kb regions (see Section 2) to predict putative bacteriocins. The w2v+RNN model predicted 119 putative bacteriocins with a probability of ≥ 0.99. Figure 9 shows three of our predicted bacteriocins in their genomic neighborhood in *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, a group of bacterial toxins which are challenging to discover using sequence similarity as they are, in many cases, non-homologous. Our approach does not require sequence similarity searches, and has discovered several putative bacteriocins with a high probability. The Word2vec representation takes advantage of

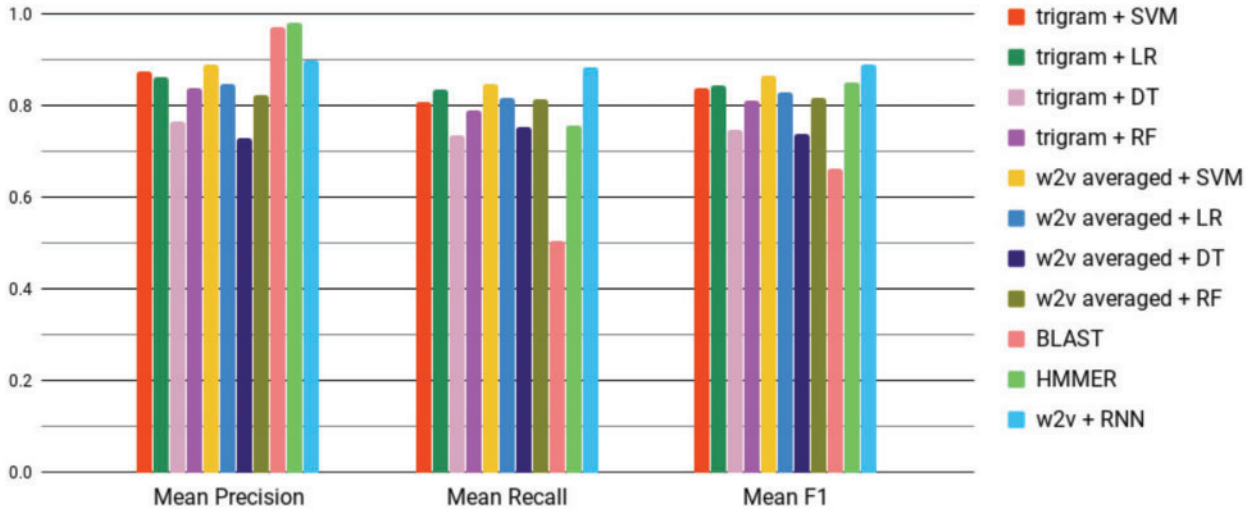

Fig. 7. Mean F_1 scores of different algorithms with both Word2vec (w2v) and baseline representations. Error bars (using standard error) are too small to be shown. W2v + RNN (blue, rightmost bar in each grouping) provides the best F_1 score. See Supplementary Table S1 for mean and standard errors values. (Color version of this figure is available at *Bioinformatics* online.)

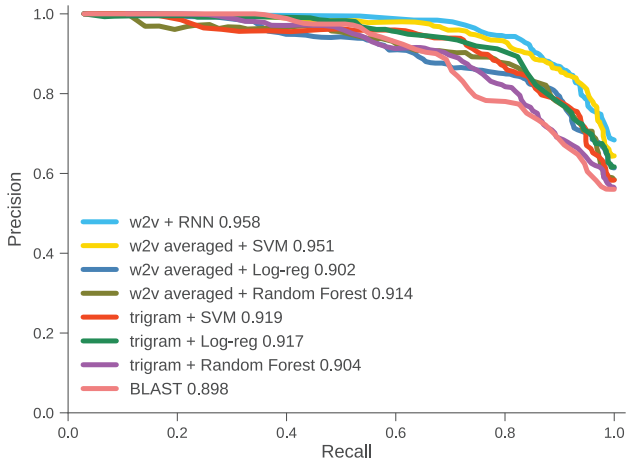

Fig. 8. Mean precision-recall curves of one run of 10× cross validation for Word2vec with RNN, Support Vector Machine (SVM), Logistic Regression (Log-reg), Random Forest and BLAST. Number in legend is area under the curve. w2v+RNN performs better than all the other methods

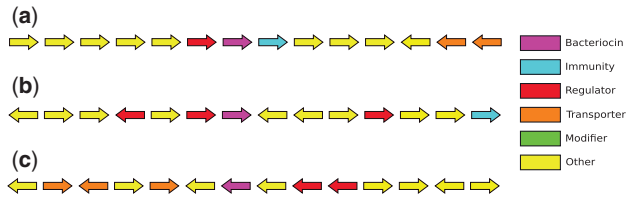

Fig. 9. Context genes found surrounding the predicted bacteriocins within ±25kb range. (**a**) *Lactobacillus acidophilus NCFM* (Locus: NC_006814, putative bacteriocin: YP_193019.1, immunity: YP_193020.1, regulator: YP_193018.1, transporters: YP_193025.1, YP_193026.1). (**b**) *Lactobacillus helveticus 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)

the large volume of unlabeled bacterial protein sequences available, and can be used in other machine learning tasks in computational biology to represent protein sequences for discovering functional similarities that cannot be discovered from sequence similarity. We used the embedding vectors for each overlapping trigram in a protein sequence, and used them as input in a temporal order for an RNN, and with heavy regularization, it performed well. We hypothesize that the unsupervised step helped transfer important information through the trigram vectors such that the RNN's task was made easier. Another reason we used RNN, is that we can represent each protein sequence as overlapping trigrams. As a result, vectors of size 200 representing each subsequent trigram can be fed into the RNN without dramatically increasing the feature space as opposed to SVM or Logistic Regression. For SVM or LR we would have needed to find another way to represent a protein sequence from its overlapping trigram vectors, so that its feature size does not overpower the number of training sequences available to us. For example, in the work by Asgari and Mofrad, (2015), the researchers summed up all the overlapping trigram vectors to represent a protein sequence. In this paper, we averaged the overlapping trigram vectors to represent a protein sequence, and used that as a baseline. We also built three different datasets using different sets of negative bacteriocin examples. All the methods except w2v+RNN and averaged w2v+SVM struggled to identify true bacteriocins in the primary bacteriocin dataset where the length distribution for positive and negative bacteriocins is exact. This is also the reason we used the primary bacteriocin dataset as the final dataset to train our RNN model before applying it to find novel bacteriocins in *Lactobacillus*. Compared with the primary bacteriocin dataset, the other methods except BLAST and HMMER have had improved performance as the differences in length distribution of positive and negative sequences increased in the second and third bacteriocin dataset. The BOA study (Morton *et al.*, 2015) supplied us with pHMMs that were built using many sequences including the BAGEL dataset, and used with HMMER. Yet tested against the BAGEL sequences, HMMER's precision is high but the recall remained low compared with w2v+RNN.

Despite the training set being small, with proper regularization our RNN model provides a better precision than all the other

methods except BLAST and HMMER, and better recall than all other methods. We argue that word embedding and RNN can be used to boost the prediction 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. We chose a threshold of 0.99 for RNN 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 in the microbial sequence universe is much lower. Finally, we provide six protein sequences that our model predicted to be bacteriocins, with a probability of ≥ 0.99, where we could also find putative context genes. We also provide a set of total 119 sequences predicted by w2v+RNN with a probability of greater than 0.99. None of these sequences could be detected against known bacteriocins when we used BLAST against the *nr* database with an e-value $\leq 10^{-3}$.

Historically, the use of bioinformatics prediction methods has favored high precision over high recall, as a large number of false positive findings can be costly for experiments that verify predictions. However, there are cases where a high recall method is appropriate. For example, with the need to cast a wider net in identifying potential drug candidates, driven by decrease drug scanning costs. By employing a high recall method and choosing an appropriate accuracy threshold, experimentalists can calibrate the precision/recall trade off needed to optimize the functional testing of novel peptides.

Protein classification tasks are typically based on some form of sequence similarity as an indicator for 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 (Verspoor *et al.*, 2012), genomic context methods (Enault *et al.*, 2005; Huynen *et al.*, 2000; Overbeek *et al.*, 1999) and other combined methods (Franceschini *et al.*, 2013). However, such methods require associated metadata or contextual genomic information. Here we present a solution to find functionally similar non-orthologs that does not require gathering these metadata, but does require a dataset of positive and negative examples. We therefore recommend that word embedding be explored for function classification involving dissimilar biological sequences.

Acknowledgements

We thank Drena Dobbs, Adina Howe and Hagit Shatkay for their input. We gratefully acknowledge all members of the Friedberg lab for stimulating discussions of this work.

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 and NSF grant: ABI-1458359. 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.

Conflict of Interest: none declared.

References

Altschul,S.F. *et al.* (1997) Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Apweiler,R. *et al.* (2004) Uniprot: the universal protein knowledgebase. *Nucleic Acids Res.*, **32**, D115–D119.

Asgari,E. and Mofrad,M.R. (2015) Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS One*, **10**, e0141287.

Benson,D.A. *et al.* (2014) Genbank. *Nucleic Acids Res.*, **42**, D32–D37.

Boutet,E. *et al.* (2016) UniProtKB/Swiss-Prot, the manually annotated section of the UniProt KnowledgeBase: how to use the entry view. In: Edwards,D. (ed.) *Plant Bioinformatics. Methods in Molecular Biology*, Vol 1374. Humana Press, New York, NY.

Centers for Disease Control and Prevention, (2013) Antibiotic Resistance Threats in the United States, 2013 (AR Threats Report). CDC 1600 Clifton Rd. Atlanta, GA, https://www.cdc.gov/drugresistance/biggest_threats.html.

Chen,Z. *et al.* (2018) Evaluating semantic relations in neural word embeddings with biomedical and general domain knowledge bases. *BMC Med. Inf. Decis. Mak.*, **18**, 53–68.

Chollet,F. *et al.* (2015) Keras. https://scholar.google.com/scholar?hl=en&as_sdt=0,16&cluster=17868569268188187229.

Cock,P.J. *et al.* (2009) Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics (Oxford, England)*, **25**, 1422–1423.

de Vos,W.M. *et al.* (1995) Maturation pathway of nisin and other lantibiotics: post-translationally modified antimicrobial peptides exported by gram-positive bacteria. *Mol. Microbiol.*, **17**, 427–437.

Du,J. *et al.* (2018) Gene2vec: distributed representation of genes based on co-expression. *bioRxiv*.

Duong,D. *et al.* (2018) Word and sentence embedding tools to measure semantic similarity of gene ontology terms by their definitions. *J. Comput. Biol.*

Dutta,A. *et al.* (2018) Splicevec: distributed feature representations for splice junction prediction. *Comput. Biol. Chem.*, **74**, 434–441.

Eddy,S.R. (2011) Accelerated profile hmm searches. *PLoS Comput. Biol.*, **7**, e1002195.

Enault,F. *et al.* (2005) Phydbac' gene function predictor': a gene annotation tool based on genomic context analysis. *BMC Bioinformatics*, **6**, 247.

Franceschini,A. *et al.* (2013) String v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res.*, **41**, D808–D815.

Fu,L. *et al.* (2012) Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, **28**, 3150–3152.

Guder,A. *et al.* (2000) Posttranslationally modified bacteriocins the lantibiotics. *Biopolymers*, **55**, 62–73.

Hammami,R. *et al.* (2010) Bactibase second release: a database and tool platform for bacteriocin characterization. *BMC Microbiol.*, **10**, 22.

Hunter,J.D. (2007) Matplotlib: a 2d graphics environment. *Comput. Sci. Eng.*, **9**, 90–95.

Huynen,M. *et al.* (2000) Predicting protein function by genomic context: quantitative evaluation and qualitative inferences. *Genome Res.*, **10**, 1204–1210.

Kingma,D. and Ba,J. (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Kluyver,T. *et al.* (2016) Jupyter notebooks-a publishing format for reproducible computational workflows. In: Fenando,L. and Birgit,S. (eds) *Proceedings of the 20th International Conference on Electronic Publishing*, ELPUB, IOS Press BV, Amsterdam, Netherlands, pp. 87–90.

Li,W. and Godzik,A. (2006) Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics (Oxford, England)*, **22**, 1658–1659.

McAuliffe,O. *et al.* (2001) Lantibiotics: structure, biosynthesis and mode of action. *FEMS Microbiol. Rev.*, **25**, 285–308.

Mejia Guerra,M.K. and Buckler,E.S. (2017) k-mer grammar uncovers maize regulatory architecture. *bioRxiv* 222927.

Mikolov,T. *et al.* (2013) Efficient estimation of word representations in vector space. *arXiv preprint arXiv: 1301.3781*.

Minarro-Giménez,J.A.A. *et al*. (2014) Exploring the application of deep learning techniques on medical text corpora. *Stud. Health Technol. Inf.*, **205**, 584–588.

Mohimani,H. *et al*. (2014) Automated genome mining of ribosomal peptide natural products. *ACS Chem. Biol.*, **9**, 1545–1551.

Mohimani,H. *et al*. (2017) Metarippquest: a peptidogenomics approach for the discovery of ribosomally synthesized and post-translationally modified peptides. *bioRxiv*, 227504.

Morton,J.T. *et al*. (2015) A large scale prediction of bacteriocin gene blocks suggests a wide functional spectrum for bacteriocins. *BMC Bioinformatics*, **16**, 381.

Overbeek,R. *et al*. (1999) The use of gene clusters to infer functional coupling. *Proc. Natl. Acad. Sci. USA*, **96**, 2896–2901.

Pedregosa,F. *et al*. (2011) Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.

Pruitt,K.D. *et al*. (2006) Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res.*, **35**, D61–D65.

Řehůřek,R. and Sojka,P. (2010) Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pp. 45–50.

Riley,M.A. and Wertz,J.E. (2002) Bacteriocins: evolution, ecology, and application. *Annu. Rev. Microbiol.*, **56**, 117–137.

Rost,B. (1999) Twilight zone of protein sequence alignments. *Protein Eng.*, **12**, 85–94.

Srivastava,N. *et al*. (2014) Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, **15**, 1929–1958.

van Heel,A.J. *et al*. (2013) Bagel3: automated identification of genes encoding bacteriocins and (non-) bactericidal posttranslationally modified peptides. *Nucleic Acids Res.*, **41**, W448–W453.

Verspoor,K.M. *et al*. (2012) Text mining improves prediction of protein functional sites. *PLoS One*, **7**, e32171. +.

Walt,S. v d. *et al*. (2011) The numpy array: a structure for efficient numerical computation. *Comput. Sci. Eng.*, **13**, 22–30.

Weber,T. *et al*. (2015) antismash 3.0 a comprehensive resource for the genome mining of biosynthetic gene clusters. *Nucleic Acids Res.*, **43**, W237–W243.

Willey,J.M. and van der Donk,W.A. (2007) Lantibiotics: peptides of diverse structure and function. *Annu. Rev. Microbiol.*, **61**, 477–501.

Zhang,R. *et al*. (2018) Predicting ctcf-mediated chromatin loops using ctcf-mp. *Bioinformatics (Oxford, England)*, **34**, i133–i141.

Zwierzyna,M. and Overington,J.P. (2017) Classification and analysis of a large collection of in vivo bioassay descriptions. *PLoS Comput. Biol.*, **13**, e1005641.