ToxVec: Deep Language Model-Based Representation Learning for Venom Peptide Classification

Meisam Ahmadi1, Mohammad Reza Jahed-Motlagh1,∗, Ehsaneddin Asgari2, Adel Torkaman Rahmani1, and Alice C. McHardy2,∗

1Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
2Computational Biology of Infection Research, Helmholtz Center for Infection Research, 38124 Braunschweig, Germany

Corresponding author:
Mohammad Reza Jahed-Motlagh and Alice McHardy1,∗
Email address: jahedmr@iust.ac.ir AND Alice.McHardy@helmholtz-hzi.de

ABSTRACT

Venom is a mixture of substances produced by a venomous organism aiming at preying, defending, or intraspecific competing resulting in certain unwanted conditions for the target organism. Venom sequences are a highly divergent class of proteins making their machine learning-based and homology-based identification challenging. Prominent applications in drug discovery and healthcare, while having scarcity of annotations in the protein databases, made automatic identification of venom an important protein informatics task. Most of the existing machine learning approaches rely on engineered features, where the predictive model is trained on top of those manually designed features. Recently, transfer learning and representation learning resulted in significant advancements in many machine learning problem settings by automatically learning the essential features. This paper proposes an approach, called ToxVec, for automatic representation learning of protein sequences for the task of venom identification. We show that pre-trained language model-based representation outperforms the existing approaches in terms of the F1 score of both positive and negative classes achieving a macro-F1 of 0.89. We also show that an ensemble classifier trained over multiple training sets constructed from multiple down-samplings of the negative class instances can substantially improve a macro-F1 score to 0.93, which is 7 percent higher than the state-of-the-art performance.

Availability: The ToxVec application is available to use at https://github.com/meahmadi/ToxVec

1 INTRODUCTION

Venom is a mixture of enzymatic or non-enzymatic substances produced by the body of a venomous organism aiming at preying, defending, or intraspecific competing (Casewell et al., 2013) resulting in immobilizing or paralyzing the target organism. Venom has evolved independently multiple times throughout the tree of life, making the evolutionary study of venom a significant interest (Jenner et al., 2019). Being rich in having ion channels, G-protein-coupled receptors, and transporters have made venom an excellent source for therapeutics and drug discoveries (Lewis and Garcia, 2003; Prashanth et al., 2017). Despite prominent applications of venom in drug discovery and healthcare, only a small portion of proteins are annotated in large protein databases (UniProt/SwissProt) to be venom (Jungo et al., 2012) (Currently, 6,736 out of 563,082 protein sequences in Swiss-Prot). This gap motivates computational methods that can automatically and accurately identify venom peptides in the large protein datasets. The prediction of venoms versus non-venom sequences is not a trivial task protein classification task, where the use of BLAST-based approaches is challenging: venoms are often (i) evolved from non-toxic proteins
contains 274 verified venom protein sequences (2016–2018, not included in the
which we could compare with our proposed ToxVec.

**ClanTox** (Naamati et al., 2009) is a machine learning-based classification of venom available as a
web-server. In the ClanTox, each sequence is encoded into a vector of 545 global sequence features
and the predictive model consisting of 10 boosted-stump classifiers is trained over the dataset of known
venoms (Iba and Langley, 1992) scoring venom on a scale of -1 to 1. **ToxClassifier** (Gacesa et al.,
2016) is an ensemble predictor using nine Support Vector Machine (SVM) (Cortes and Vapnik, 1995),
Gradient Boosted Machine (GBM)(Friedman, 2002) and Generalised Linear Model (GLM) (Nelder
and Wedderburn, 1972) classifiers over different combinations of features including sequence length,
frequency of amino acids, amino acid dimer frequency, Hidden Markov Models (HMM) of tox-bit motifs
(Starcevic et al., 2015), homology-based features (against a positive venom database). **Toxify** (Cole and
Brewer, 2019) is a deep learning-based venom predictor employing Recurrent Neural Networks (RNN)
and, in particular, the Gated Recurrent Units (GRUs) variation of RNN (Cho et al., 2014) for sequence
modeling and ultimately prediction. For sequence encoding, toxify uses five Atchley factors per amino
acid in the protein (Atchley et al., 2005). Similarly, in this paper, we propose a deep-learning approach for
supervised training of the venom predictor model. However, instead of using manually extracted features,
we propose a transfer learning framework. Similar to ProtVec (Asgari and Mofrad, 2015) and ProtVecX
(Asgari, 2019; Asgari et al., 2019a), we use a skip-gram network (Bojanowski et al., 2017; Mikolov et al.,
2013) which is analogous to language modeling. Subsequently, the pretrained network is fine-tuned for
the venom classification task.

Recently, transfer learning resulted in significant advancements in many machine learning problem
settings, particularly for inadequately annotated data (Bengio, 2012; Tan et al., 2018; Wolf et al., 2019).
Transfer learning in machine learning refers to the use of the solution in a problem setting (source problem)
with enough training samples/prior knowledge to solve a different problem (target problem) with less
training samples/prior knowledge. Using a neural network trained relevant representations for a specific
task for another task is also an instance of transfer learning through representation learning (Bengio, 2012;
Tan et al., 2018). Combinations of being self-supervised and being general enough make neural language
modeling an ideal candidate for transfer learning. Afterward, the trained language modeling network can be fine-tuned for any particular task, even when
only a limited number of annotations are available. Here we describe the use of Skip-gram (Bojanowski et al.,
2017; Mikolov et al., 2013), one of the most successful architecture to perform transfer learning on
natural language text for the task of venom prediction.

This paper shows that fine-tuning of language model-based representation outperforms the state-of-
the-art approaches in venom peptide classification. In addition, ensemble classifiers trained on resamples
of negative samples (the major class) further improve the macro-F1 of both negative and positive classes.

**METHODS**

**1.1 Datasets**

For the ease of benchmarking, we use the dataset created and proposed by Toxify (Cole and Brewer, 2019)
containing training and test protein sequences:

The Toxify training dataset contains (i) **Positive examples**: 6,133 venom protein sequences extracted
from Swiss-Prot sequences annotated with **annotation:**(type: tissue specificity venom)), (i) **Negative
examples**: 50,000 random protein sequences from Swiss-Prot satisfying the query **NOT annotation:**(type:
tissue specificity venom), these sequences only include the sequences uploaded prior to June 2016 on
Swiss-Prot.

The Toxify test dataset contains 274 verified venom protein sequences (2016–2018, not included in the
training) and 94 verified non-venom protein sequences from the same time interval of (2016–2018).
1.2 Skip-gram analogous to Language Modeling

Language modeling aims to assign a probability \( P(w_1, w_2, \ldots, w_N) \) to a given sequence of elements (words, phrases, or amino acids in proteins) \( w_1, w_2, \ldots, w_N \). Language modeling is a vital component in many language processing applications, particularly the applications containing language generation or the evaluation of text correctness, e.g., chat-bot or machine translation. Language modeling probability can be written as follows using the chain rule:

\[
P(w_1, w_2, \ldots, w_N) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1, w_2) \times \ldots \times P(w_N|w_1, \ldots, w_{N-1})
\]

Requiring only raw data and being general enough has made language modeling a favorable task for transfer learning. Recently, transfer learning from the language modeling became a very popular method in natural language processing and bioinformatics and obtained state-of-the-art performance in many tasks (Asgari and Mofrad, 2015; Bengio, 2012; Howard and Ruder, 2018; Rao et al., 2019; Tan et al., 2018). A variety of language models are proposed in the literature. In this paper, we focus on Skip-gram neural network (depicted in Figure 1.1) whose objective is analogous to the objective of the language modeling task. However, in skip-gram the input and output are swapped and it predicts the surroundings (context) for a given textual unit. The objective of skip-gram is to maximize the following log-likelihood:

\[
\sum_{t=1}^{M} \sum_{c \in [t-N,t+N]} \log p(w_c | w_t),
\]

where \( N \) is the surrounding window size around word \( w_t \), \( c \) is the context indices around index \( t \), and \( M \) is the corpus size in terms of the number of available words and context pairs. This probability of observing a context word \( w_c \) given \( w_t \) is parameterized using word embedding:

\[
p(w_c | w_t; \theta) = \frac{e^{w_c \cdot v_t}}{\sum_{c' \in C} e^{w_{c'} \cdot v_t}},
\]

where \( C \) denotes all existing contexts in the training data. However, iterating over all existing contexts is computationally expensive. This issue can be efficiently addressed by using negative sampling. In a negative sampling framework, we can rewrite Equation 1 as follows:

\[
\sum_{t=1}^{T} \left[ \sum_{c \in [t-N,t+N]} \log \left( 1 + e^{-s(w_t, w_c)} \right) + \sum_{w_{t} \in \mathcal{N}_c} \log \left( 1 + e^{s(w_t, w_{t})} \right) \right],
\]

where \( \mathcal{N}_c \) denotes a set of randomly selected negative examples sampled from the vocabulary collection as non-contexts of \( w_t \) and \( s(w_t, w_c) = v_t^\top \cdot v_c \) (parameterization with the word vector \( v_t \) and the context vector \( v_c \)) (Goldberg and Levy, 2014). The use of Skip-gram for protein sequences and transfer learning in protein informatics has been proposed by a number of recent works (Asgari et al., 2019a; Asgari and Mofrad, 2015; Wan and Zeng, 2016).

1.3 Overview of Approach

Here we describe our approach ToxVec in the use of language-model based representation for the classification of venom peptides. The ToxVec computational workflow has the following steps (as depicted in Figure 1):

1. Unsupervised Training of the Language Model-based Embeddings: In this step (Figure 1.1), we train a protein k-mer representation proposed in (Asgari and Mofrad, 2015), ProtVec. For this study, we used a recent version of ProtVec where the training is expanded from Swiss-Prot (containing \( \approx 500K \) sequences) to a much larger set, UniRef90, containing \( \approx 115M \) protein sequences. Next, the protein sequences are divided into non-overlapping 3-mers by adding two starting symbols of @@ and two ending symbols of @ @. As detailed in (Asgari and Mofrad, 2015) and also shown in Figure 1, all three ways of splitting (based on the starting position for splitting) is done (i) to increase the training size to
\[ 115M \times 3 = 445M \] sequences of k-mers and (ii) to capture all possible neighborhoods. The skip-gram network is trained on the mentioned collection of divided sequences, with the window size of 20, and the vector size of 3000.

Figure 1. Overview of the ToxVec approach for the detection of venom proteins using fine-tuning of language model-based representations. The steps are detailed in the §1.3. (1) The first step is the training of Skip-gram embedding for protein k-mers over UniRef90, (2) We draw multiple (N=10) resamples from the major class (negative set), (3) We fine-tune the Skip-gram embeddings for the venom classification in the classification network, (4) The eventual output is the aggregated result from (\( N = 10 \) classifiers).

2. Multiple Resampling from the Major Class (negative) Since in the training dataset provided by (Cole and Brewer, 2019), the negative set is almost eight times larger than the positive set, the classifier is subject to be biased towards the negative class. To address this issue, we downsample the negative set to the positive set’s size to mitigate this bias. In addition, next, to ensure the use of more negative samples, we perform N resamplings of the negative set and subsequently train N classifiers (N=10).

3. Supervised Fine-tuning of Embeddings for the Venom Classification For each resampled training set (in step 2), we train a classification network in the next step. As classification model, we used the fasttext model (Bojanowski et al., 2017), a simple but effective model for sentence classification in NLP: the input embeddings (here k-mer embedding) are averaged followed by a feedforward layer before the ending sigmoid layer produces the class conditional probabilities. For the k-mer embedding of the input sequences, we use the ProtVec embeddings detailed in the 1st step. We fine-tune the k-mers embedding in the course of supervised training. To investigate the role of pretrained ProtVec in classification performance, we repeat the same experiment with randomly initialized k-mer embedding.

Furthermore, since in the creation of embedding training corpus (step 1), each protein sequence is divided into three sequences of k-mers (k=3), the test set sequences would also undergo the same procedure. Thus, at the inference time, for each test sequence, we would have three possible segmentations (e.g., esecpwhpwc \( \rightarrow \) (1) ##e, sec, pwh, pwc (2) #e, ecp, whp, wc@ (3) ese, cpw, hpw, c@@) and subsequently we would have three classification outcomes. This way, we have three binary outcomes for
Figure 2. Distribution of biophysical and biochemical properties in the protein trimers (except (f)) and in venom sequences versus non-venoms (f) in the embedding space visualized using t-SNE. The five heatmaps scatter plots of biophysical properties (Figures (a) to (e)) show the standardized scales averaged for each trimer. Figure (f) shows the distribution of training instances of venom (colored in red) versus non-venom (colored in blue) in this space.

Figure 3. Visualization of test (a), train (f), and misclassified instances ((b) to (e)) using different existing venom predictive models in the embedding space of protein k-mers (trimers) for venom prediction. The ToxVec (b), Toxify (c), ClanTox (d), ToxClassifier (f), and ToxVec misclassified instances are compared. In Figures (a) and (f), the red points are venom sequences and the blue points indicate the non-venom sequences. In Figures (b) to (e), the red points indicate the venom sequences identified as non-venom by the predictor and the blue points are the non-venom sequences identified as venom sequences.

156 each protein sequence in the test set, and we assign the majority class for each sequence.
4. The ensemble classifier of different resamples As discussed in step 2, we create $N = 10$ training sets resulting in 10 predictive models. We consider a positive sample for the eventual classification output if and only if all 10 models confirm this.

## 2 RESULTS

Venom Protein classification results over the Toxify test set for ToxVec, Toxify, ToxClassifier, and ClanTox are provided in Table 1. For the evaluation, the accuracy, the F1 score of positive and negative classes, and their average (macro-F1) are reported. Our ToxVec outperformed ClanTox, ToxClassifier, and Toxify in terms of F1 on both positive and negative class by improving macro-F1 (average of F1 on positive and negative classes) for 3 percent from 0.86 to 0.89. Furthermore, the incorporation of negative-set resamplings increased the performance to a macro-F1 of 0.93.

### Table 1. The summary of evaluation results for detecting venom proteins in the Toxify test set: We compare the performance of where ToxVec and its ensemble version with ClanTox, ToxClassifier, and Toxify approaches in terms of accuracy, $F_1$s of both positive and negative classes, and the macro-average of $F_1$s. The performance of ToxVec for both initialization modes (random initialization and ProtVec-based initialization) are provided.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>F1-positive</th>
<th>F1-negative</th>
<th>macro-F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClanTox</td>
<td>0.79</td>
<td>0.84</td>
<td>0.69</td>
<td>0.77</td>
</tr>
<tr>
<td>ToxClassifier</td>
<td>0.73</td>
<td>0.78</td>
<td>0.65</td>
<td>0.72</td>
</tr>
<tr>
<td>Toxify</td>
<td>0.86</td>
<td>0.85</td>
<td>0.87</td>
<td>0.86</td>
</tr>
<tr>
<td>ToxVec(Random – init – Emb)</td>
<td>0.9</td>
<td>0.86</td>
<td>0.93</td>
<td>0.92</td>
</tr>
<tr>
<td>ToxVec – Ensembled(Random – init – Emb)</td>
<td>0.94</td>
<td>0.87</td>
<td>0.96</td>
<td>0.92</td>
</tr>
<tr>
<td>ToxVec(UniRef90 – Emb)</td>
<td>0.91</td>
<td>0.83</td>
<td>0.94</td>
<td>0.89</td>
</tr>
<tr>
<td>ToxVec – Ensembled(UniRef90 – Emb)</td>
<td>0.95</td>
<td>0.89</td>
<td>0.96</td>
<td>0.93</td>
</tr>
</tbody>
</table>

We created a t-SNE (Maaten and Hinton, 2008) visualization of the Skip-gram embedding space of protein trimers (Figure 2). In this figure, the trimers of vector size 3000 are mapped into a 2D space. Next, to see how biophysical properties are distributed in this embedding space we color the k-mers for different properties, including mean molecular weight, mean surface accessibility (Emini et al., 1985), mean KD hydrophilicity (Kyte and Doolittle, 1982), mean flexibility (Vihinen et al., 1994), and mean Janin Interior to surface transfer energy scale (Janin et al., 1988). The mentioned biophysical scales are standardized (zero mean and unit variance) to be comparable. Higher intensity (lighter color) indicates being higher in the scales. We can see that the k-mers of similar properties are close in the embedding space. Afterward, we represent Toxify’s training instances with the average of their overlapping timers and then mapped them to the 2D space using the same t-SNE projection of simple trimers. The bottom-right sub-figure in Figure 2 shows the venom sequences in red and the non-venoms in blue. Comparison of training instances and the biophysical properties shows the average properties of typical venom sequences versus non-venom protein sequences. The illustration shows that the venoms are diverse in terms of averaged biophysical properties, which is confirmed previously even within certain snake families (Nawarak et al., 2003).

## CONCLUSIONS AND DISCUSSIONS

Here, we described ToxVec, a deep learning model using language model-based representation learning of proteins for venom protein identification. We compared the performance of ToxVec with recent supervised approaches in venom identification and showed that the supervised fine-tuning of protein language model-based representation achieved state-of-the-art performance in this task. We also addressed the class-imbalance problem in training a predictive model by ensembling models trained on the major class’s downsampling, further improving the performance by 4 percent macro F1 (a macro-F1 of 0.93).

Figure 3 showed the visualization of test cases (a), train cases (f, and the misclassified instances using different approaches. The figure suggests that the test cases were not similar to the typical training instances, and the problem has not been trivial for the embedding space. The misclassified instances of
Toxify, ToxC Classifier, and ClanTox follow the same patterns. When ToxVec was employed, the F1 scores on both venoms/non-venoms classes were improved, which was even better in the negative class.

We observed that the ToxVec outperformed the state-of-the-art venom predictors by 2% to 7% macro-F1 (averaged F1 in the positive and negative class). The minimum macro-F1 of ToxVec, 0.88, which was still higher than existing approaches macro-F1 (0.86), was achieved when an embedding layer was trained for k-mers from scratch in a supervised manner. By ensembling ten classifiers trained on different downsampling of the negative set, this performance increased to a macro-F1 of 0.92. We also showed that when the pretrained Skip-grams over UniRed are used, the macro-F1 and all scores are increased by one more point (macro-F1 = 0.93). These results suggest that automatic feature learning, either by random initialization and then supervised training or fine-tuning of self-supervised embedding, can improve venom identification performance compared to methods using manual feature engineering. Like natural language processing scenarios, fine-tuning of language model-based representations improved the downstream supervised task performance, which is particularly evident for small training sets. The success of automatic representation learning approaches in our experiments motivates exploring of contextualized embedding (transformers (Rao et al., 2019) or ELMo embeddings (Asgari et al., 2019b; Heinzinger et al., 2019)) as future directions.

REFERENCES


