DeepMSim: A Deep Learning based Simulator for MS/MS Spectra Data

Muhammad Haseeb, and Sumesh Kumar

***Abstract*—In computational proteomics, the peptide molecules in a complex biological mixture are identified by searching their corresponding experimental mass-spectrometry spectra (discrete 2D signal-like data) against a database of simulated MS/MS spectra data. Therefore, it is of prime importance that the theoretically predicted mass-spectrometry spectra closely match to their corresponding experimental spectra. In this work, called DeepMSim, we intend to learn the *fragmentation pattern* of peptides when they are subjected to mass-spectrometry under certain conditions so that if we are given a peptide molecule, we could theoretically predict its (expected) spectrum. Our DeepMSim is based on a Transformer network which is fed a sequence of characters that represent a peptide molecule and the model outputs a sequence of fragments that make the theoretical spectrum of that input peptide molecule. The novel feature about DeepMSim is that it also predicts the *immonium ions* in addition to the traditionally predicted b/y (terminal) ions in its output sequence. The results show about 27% and 36% prediction accuracy for the NIST mouse and human HCD datasets containing about 50K and 500K training examples respectively. The results depict that there is significant margin of improvement in the proposed method which is a part of our future work.**

***Index Terms*—deep learning, proteomics, mass-spectrometry, transformers**

I. Introduction

IDENTIFYING peptide sequence labels from the MS/MS spectra (2D discrete signal-like) data is a highly challenging task in computational proteomics. The most commonly employed method for sequencing (assigning a sequence label to a MS/MS spectrum) is called the database peptide search. The database peptide search involves comparing the experimentally obtained mass-spectrometry data against the theoretically simulated mass-spectrometry data [1] as shown in Fig.1. Therefore, it is extremely important that the data simulation is accurate and close to the experimental data.

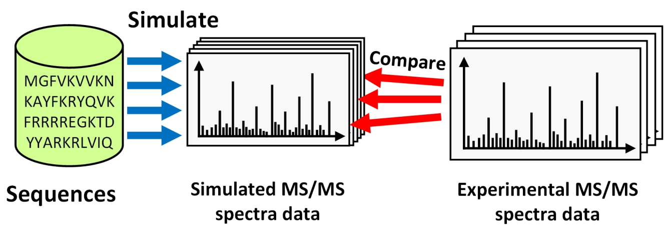


Fig. 1. The simulated data is generated by peptide sequences (string data). Experimental data is compared with simulated data for identification.

M. Haseeb and S. Kumar are with School of Computing and Information Sciences, Florida International University, Miami, FL, 33199 USA email:{mhase003,skuma027}@fiu.edu

A peptide molecule consists of long chains of amino acid molecules bound to each other by what's called a *peptide bond*. There are a total of 20 amino acids and all peptides are formed by different permutations and lengths of these amino acids. Each amino acid is represented by an alphabet of English language except for the characters B, J, O, W, X, Z. When a peptide molecule undergoes mass-spectrometry, one or more bonds break forming smaller fragments forming a fingerprint for each molecule. Generally, there are several molecules of the same peptide that undergo fragmentation together and break at different *bond sites* or simply sites producing different fragments. The peptide molecules tend to have a fragmentation pattern under fixed conditions. i.e. same peptide molecules tend to break at same sites under same conditions producing a fragmentation-pattern. Further, we classify the fragments into three types: terminal ions, immonium ions and internal ions which are defined as:

- **Terminal Ions:** The ions that have one terminal of the peptide chain. For instance, thes ion labelled as: MAK or YQRP in the peptide sequence: MAKITYQRP are terminal ions.
- **Immonium Ions:** The ions that are made up of only one amino acid in the peptide chain. For instance, the ions labelled as M, K, A, Q in the sequence: MAKITYQRP are immonium ions.
- **Internal Ions:** The ions that are made up by two cleavages (break points) in the peptide chain and not having any terminal. For instance, the ions labelled as AKIT, or TYQR in the peptide chain: MAKITYQRP are internal ions.

Traditionally, the mass-spectrometry simulators MassAnalyzer [2], MaSS-Simulator [3] as well as [4], [5] use deductive probabilistic models for simulation. The probabilities for each ion type are set manually based on the expert opinions. We argue that instead of the expert opinion, the fragmentation patterns must be learned by the data itself for given experimental conditions for accurate mass-spectrometry data simulation. With the advent of machine learning and deep-learning, there have been multiple efforts [6], [7], [8] towards building models that learn the fragmentation patterns from the mass-spectrometry data itself. Here we present an artificial neural network (ANN) based deep learning model, called *DeepMSim*, that learns the immonium ion fragmentation patterns in addition to the terminal ions that are predicted by the state-of-the-art simulators. Our DeepMSim is based on Transformer network [9] which have been designed for mainly sequence transduction [10] problems. Therefore, we model our problem as a language translation problem where the input is sentence in language-A (amino acids in peptide sequences) and the output is a sentence in language-B (immonium ion labels). We show about 27% and 36% accuracy for NIST Mouse and Human HCD datasets containing about 50K and 500K training data points. The results depict that our model is capable of learning the fragmentation patterns and there is a significant room for improvement which is a subject of our future work.

II. Related Work

A. Sequence Transduction

In machine learning, sequence transduction is the process of learning the transformation function between an input sequence and an output sequence which is used in the applications of speech recognition and language translation problems. In a typical sequence transduction problem, a model is trained to process an input sequence to predict a corresponding output sequence e.g. to translate a sentence in French into its corresponding translated sentence in English.

B. Sequence Transduction Models

Recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have long been the popular choice of models for neural machine translation systems. RNNs take into account the position of input and output sequences by generating a sequence of hidden states, where each new hidden state is computed from the previous hidden state and the current input. LSTM units are composed of cells that contain an input gate, output gate and forget gate. Unlike RNNs where the forget rate is exponential (the farther the past input, the smaller its effect), LSTMs can vary the forget rates in order to encode both the past inputs and future inputs (bi-directional LSTMs) when generating an output. However, this sequential computational model for LSTMs becomes impractical for long sequence lengths of large datasets [9].

C. Existing ML/DL based simulators

With the advent of machine learning and deep learning, multiple works have proposed learning-based models such as [11] that use a inductive Bayesian decision tree based model to improve simulation. PeptideART [6] employed a feed forward neural network approach that resulted in better simulation results. MS^{2}PIP uses a random forest approach based regression on preprocessed data with additional features as compared to PeptideART and shows even better results. The pDeep [8] builds a BiLSTM based model for MS/MS simulation and outperforms all other tools.

III. Methods

In the following sub-sections we discuss in details the methods, algorithms and the techniques employed by the DeepMSim.

A. Deep Learning Model

We employed the Transformer [9] model for our application. Tranformer models in contrast to RNNs and LSTMs are made up of a series of encoders and decoder layers. Each encoder layer consists of multi-headed attention units followed by a dense layer. The input sentences are passed through the encoder layers which learn the positional dependencies of input words by constructing a representation of each input word in terms of all other words (called *self-attention*) and therefore finding the closely related words for encoding purpose [12], [9]. The encoder outputs are then fed to the series of decoder layers which have a similar architecture as the encoder layers. The decoder layers output words in the target language based on the previous output and the input from all encoder layers in order to find the related information from the input words. The advantage if transformer networks is that it can work in parallel similar to CNNs since the word representations can be learned in parallel and therefore can be used to consider information from longer sequences than both RNNs and LSTMs.

B. Data Acquisition and Extraction

The annotated spectral libraries containing examples of peptide sequences and their fragmentation patterns were downloaded from NIST [13]. In our codebase, we used the NIST Mouse and Human HCD spectral libraries. The mouse library contained about 50K examples whereas human library contained about 500K examples. We parsed the spectral library files (in MSP format) and constructed a dictionary of all ions seen in the library constructing a dataframe storing information about the peptide sequence, charge, mass-spectrometer's energy, and the generated ions for each example in the library.

C. Data Preprocessing

The ions generated by a peptide sequence are typically sorted based on their molecular mass and not their cleavage point (generation point) in their parent peptide sequence. Therefore, in order to formulate the problem as translation problem, we aligned the generated ions to their peptide sequences so that a 1 to 1 mapping could be constructed. We then constructed the sentences for each language where the source language was constructed as:

$$lan(src) = z\ AA\ eV\ -$$

Here z represents the charge on the peptide sequence AA represents the *space* separated amino acid sequence of peptide and eV represents the mass-spectrometry energy in electron Volts. The output or target language was constructed as:

$$lan(tar) = ions-$$

where $ions$ represents the *space* separated ion codes.

D. Tokenization and Position Encoding

We then used the TensorFlow's $SubwordTextEncoder$ module to convert the sentences from the two example languages (corpuses) into tokenized form by learning tokens from the either languages as well. Further, padding and masking was added in order to create equal sized outputs for tensors. Further, a masking mechanism was used in order to avoid the transformers to perform attention on the padded inputs.

Then, since the transformer model doesn't contain any recurrence or convolution, positional encoding is added to the tokenized examples (embeddings) of each language to give our transformer information about the relative position of the words in the sentence. We do this by adding the positional encoding vector to the embedding vector (tokens vector). The embedding vectors by themselves do not encode the relative position of words in a sentence such that the words in each language will be closer to each other based on the similarity of their meaning and their position in the sentence, in the d-dimensional space [14].

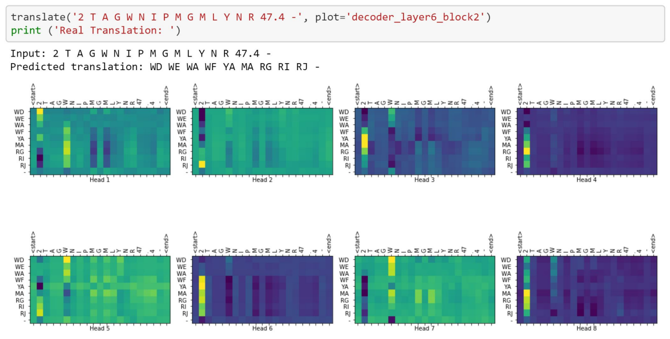
E. Training

We trained our Transformer model on both Human and Mouse datasets by creating 80/20 train/test splits in cross-validation training manner. The transformer model implementation was directly taken from the original Transformer implementation by [9] in Tensor2Tensor library which is a part of TensorFlow. For our project, we modified the code from Tensorflow's tutorial on language translation using Transformers [14]. We tuned several hyperparameters based on the output accuracy of our simulator and the best Transformer network hyperparameters were found to be the ones similar to the original paper [9] i.e number of layers = 6, dimension of vector = 128, dff = 512 and num of heads = 8. The learning rate and function used was an Adam optimizer with the formula as described in the original paper[9].

IV. Results

We trained our Transformer model on both Mouse and Human spectral library using the 80/20 train/test split using cross-validation scheme. We ran 20 epochs on the mouse dataset while only 2 epochs on the human dataset saturated the model accuracy since the human dataset was very huge (about 500K examples). The test accuracy obtained for Mouse dataset after running all 20 epochs was *27.05%* whereas the test accuracy for Humans dataset after 2 epochs was *36.93%* accuracy. We were able to further remove the false positive ions (words) in our output sequence by removing any ions for which there is no cleavage site (or corresponding amino acid character) in the input peptide sequence.

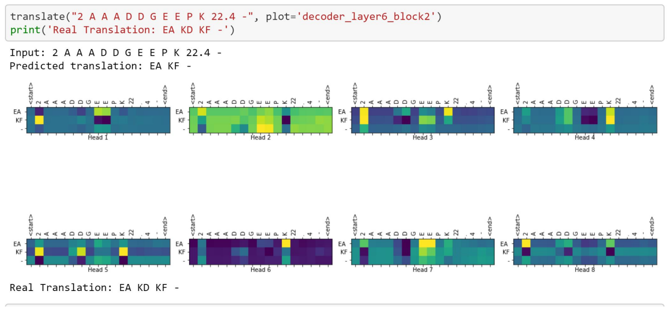
We noticed an interesting thing in a few examples that the that the experimental conditions (i.e. charge and the energy information) encoded in the peptide sequence sentence in the input language had significant impact on the correctness of the generated ion sequences as shown in Figure 2. However, in other cases, the output was highly correlated to the corresponding amino acid characters in the input (Figure 3). This observation from the output demands that the energy and charge information about the peptides must learned by transformer model explicitly and should not a part of the input peptide encodings.



```
translate('2 T A G W N I P M G M L Y N R 47.4 -', plot='decoder_layer6_block2')
print ('Real Translation: ')
```

```
Input: 2 T A G W N I P M G M L Y N R 47.4 -
Predicted translation: WD WE WA WF YA MA RG RI RJ -
```

Fig. 2. The final layer decoder layer attention heads show that the output words are strongly related to the charge information in the input sequence.



```
translate("2 A A A D D G E E P K 22.4 -", plot='decoder_layer6_block2')
print('Real Translation: EA KD KF -')
```

```
Input: 2 A A A D D G E E P K 22.4 -
Predicted translation: EA KF -
```

```
Real Translation: EA KD KF -
```

Fig. 3. The final layer decoder layer attention heads show that the output words are only related to the corresponding amino acids in the input sequence.

V. Future Work

We plan to improve the accuracy of our model by encoding the charge and mass-spectrometer's energy information in the final dense layer of the model instead of encoding it as words in the input peptide sequence. Further, we intend to extend our model to output internal ions as well.

VI. Conclusion

Learning the fragmentation patterns of peptide molecules in a mass-spectrometer is an important problem in proteomics research. In this work, we present a transformer networks based simulator, DeepMSim, that models the peptide sequence to fragmentation translation problem into language translation problem and employs transformer model based ANN to perform translations using NIST human and mouse spectral library datasets. The accuracy achieved for mouse and human datasets was about 27% and 36% respectively. The results show that there is a significant room for improvement in the accuracy.

References

[1] J. K. Eng, B. C. Searle, K. R. Clauser, and D. L. Tabb, "A face in the crowd: recognizing peptides through database search," *Molecular & Cellular Proteomics*, pp. mcp–R111, 2011.

[2] Z. Zhang, "Prediction of low-energy collision-induced dissociation spectra of peptides," *Analytical chemistry*, vol. 76, no. 14, pp. 3908–3922, 2004.

[3] M. G. Awan and F. Saeed, "Mass-simulator: A highly configurable ms/ms simulator for generating test datasets for big data algorithms.," *bioRxiv*, p. 302489, 2018.

[4] J. E. Elias, F. D. Gibbons, O. D. King, F. P. Roth, and S. P. Gygi, "Intensity-based protein identification by machine learning from a library of tandem mass spectra," *Nature biotechnology*, vol. 22, no. 2, p. 214, 2004.

[5] D. L. Tabb, C. G. Fernando, and M. C. Chambers, "Myrimatch: highly accurate tandem mass spectral peptide identification by multivariate hypergeometric analysis," *Journal of proteome research*, vol. 6, no. 2, pp. 654–661, 2007.

[6] R. J. Arnold, N. Jayasankar, D. Aggarwal, H. Tang, and P. Radivojac, "A machine learning approach to predicting peptide fragmentation spectra," in *Biocomputing 2006*, pp. 219–230, World Scientific, 2006.

[7] S. Degroeve and L. Martens, "Ms2pip: a tool for ms/ms peak intensity prediction," *Bioinformatics*, vol. 29, no. 24, pp. 3199–3203, 2013.

[8] X.-X. Zhou, W.-F. Zeng, H. Chi, C. Luo, C. Liu, J. Zhan, S.-M. He, and Z. Zhang, "pdeep: Predicting ms/ms spectra of peptides with deep learning," *Analytical chemistry*, vol. 89, no. 23, pp. 12690–12697, 2017.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[10] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[11] C. Zhou, L. D. Bowler, and J. Feng, "A machine learning approach to explore the spectra intensity pattern of peptides using tandem mass spectrometry data," *BMC bioinformatics*, vol. 9, no. 1, p. 325, 2008.

[12] Jalammar, *The Illustrated Transformer*, 2019.

[13] NIST, *NIST Spectral Libraries*, 2019.

[14] TensorFlow, *Transformer model for language understanding*, 2019.