On the Use of Convolutional Neural Network for the Detection of Cyber Sexual Predation

Arjay P. Mangabo
West Visayas State University
La Paz, Iloilo City
Philippines
arjay.mangabo@wvsu.edu.ph

Frank I. Elijorde
West Visayas State University
La Paz, Iloilo City
Philippines
felijorde@wvsu.edu.ph

Bobby D. Gerardo
West Visayas State University
La Paz, Iloilo City
Philippines
bgerardo@wvsu.edu.ph

Regin A. Cabacas
West Visayas State University
La Paz, Iloilo City
Philippines
rcabacas@wvsu.edu.ph

In-Ho Ra
Kunsan National University
Miryong-dong, Gunsan
South Korea
ihra@kunsan.ac.kr

ABSTRACT

The advent of technology paved the way in making communication even faster. However, together with its advancement are the drawbacks being experienced by the majority of the netizens, one of which is the proliferation of cyber sexual predation. Studies unveiled that there are reports and investigations concerning cyber sexual predation have dramatically increased, with one out of five youths have been exposed to unwanted sexual content not only abroad but within the Philippines as well. To addressed the said concern, the researcher developed and applied machine learning algorithm to detect the occurrence of cyber sexual predation. The data extracted from perverted-justice.com and convolutional neural network were used. The model's accuracy was evaluated and has achieved a score of 80%. Also, the time complexity of the model was discussed and determined using the Big O Notation.

KEYWORDS

Cybersexual predation, Machine Learning, Convolutional Neural Network

1 INTRODUCTION

The use of intelligent devices and the Internet continue to escalate locally and abroad as it becomes a primary tool for communication nowadays [1]. The United Nations Telecommunications agency announced that 3.9 billion or 51.2% of the world's total population are already connected to the Internet by 2018 and continuously increasing at an unprecedented rate. Undoubtedly, the Internet has provided us with numerous opportunities and has been an excellent source of information. However, [5] stated that the Internet is primarily an unregulated place wherein this can put us, specifically the youth, at risk of dangers such as unwanted online solicitation.

It was revealed that reports and investigations concerning cyber sexual predation have dramatically increased, with one out of five youths have been exposed to unwanted sexual content. These cybercrimes can seriously traumatize the victims, known as cyber-trauma [7]. Currently, studies regarding the detection of cyber sexual predation in online settings remain primarily untapped. This is because of the various methods used by the perpetrators and the unavailability of data. Also, the development of tools that can intelligently detect cyber sexual predators in online conversations is deemed necessary [3].

This paper proposes an application of a method in determining predatory messages that transpire on online chat systems. Such method is meant to warn Internet users, social media platform owners, and law enforcement of the possibility that online chatter is doing something illegal. Moreover, it aims to detect the occurrence of cyber sexual predation and address its societal challenge. Deep learning algorithm Convolutional Neural Network (CNN) is used as a tool for data analysis. Furthermore, analyzed messages has been classified based on their polarity (positive and negative.

The remainder of this paper is structured as follows; section 2 provides experimental and computational details of the presented work. Section 3 presents results and discusses the findings of the study and lastly, section 4 concludes this work.

2 EXPERIMENTAL AND COMPUTATIONAL DETAILS

2.1 Sources of Data

Due to the minimal number of studies concerning the detection of cyber sexual predation on the virtual world, particularly on social media sites/applications, there had been no well-established datasets to test different methods. The datasets used in literature differ essentially on various features such as sizes, the source, and the data format. With this, this work decided to use the dataset utilized by [6], which was utilized during the PAN '12 competition: the perverted-justice.com. The website is where the transcripts are published between the identified sexual predators and volunteers posing as children or teenagers. The study in [4] built a collection and emphasized these observations on the data from the perverted-justice.com: I) the predator/interaction (victim), which were subdivided into (Ia) Predator/Victim (underage) and (Ib) Predator/Pseudo-victim (volunteer posing as a child).

2.2 Procedure

2.2.1 Data Collection. Inspired by the study of [8], similar preprocessing on the perverted-justice dataset has been done. Specifically, unrelated words like hyperlinks, user indicators, dates of transcripts, and non-alphanumeric tokens needs to be removed. After such, a term-compression method was needed to ensure that there will be no more than two consecutive existences of character in a word. Fig. 1 shows the data collection process utilized in this study. A messaging platform was utilized to gather all corpuses and is evaluated whether such message transcript is predatory or not. Secondly, a messaging download filter is present in the process in order to somehow polished the data before it undergoes the preparation phase. Lastly, the corpuses will then be downloaded as csv file which is a file type compatible and needed for the next phases.

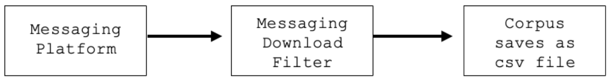


Figure 1: Data Collection Process.

2.2.2 Data Preparation. It is observed from previous studies that, especially in the domain of cyberbullying and cyber sexual predation, predators tend to use words that are slang and are intentionally misspelled, either to insult or to hide the boldness of the words [2]. The process of converting words into pronunciation will help map some misspelled words to the pronunciation of the correct one. Unfortunately, this process calso create noise such that mapping a bad word like "cum" and a common word "come" to the exact phonetic representation. However, authors in [8] believed that the benefits of the process outweigh the costs since it often maps words correctly rather than create noise.

2.2.3 Data Modeling. Table 1 shows the different attributes used for classification and the application dependent nature. The test data is also similar to the training data without the class column, which has been predicted with the use of the algorithm implementation describe in the next subsection.

Table 1: Training data for predicting cyber predation

<u>ID</u>	<u>Message</u>	<u>Sentiment</u>	<u>Class</u>
Each message corpus will have its own ID for identifying its own sentiment.	Each user's message corpus from the raw data.	Sentiment polarity of 0 and 1	Positive and Negative

2.3 System Architecture

This study utilized machine learning techniques, particularly Convolutional Neural Network (CNN) to determine the occurrence of sexual predation that is happening in cyberspace, thus, alleviating the efforts made by law enforcement to address such and to alarm internet users regarding the proliferation of the said cybercrime and lessen the possibility of becoming a victim. Google's word to vector converter has been utilized. Fig. 2 below shows the architectural design of the proposed work. This serves as the backbone of the entire model in determining the occurrence of cyber sexual predation from downloaded corpuses. Provided that this study focuses on the area of natural language processing, a one (1) dimensional convolutional neural network is used. It can be seen from the figure below that a convolutional filter and max pooling with the size of 3 is used in order to generate feature vector in predicting predatory and non-predatory messages.

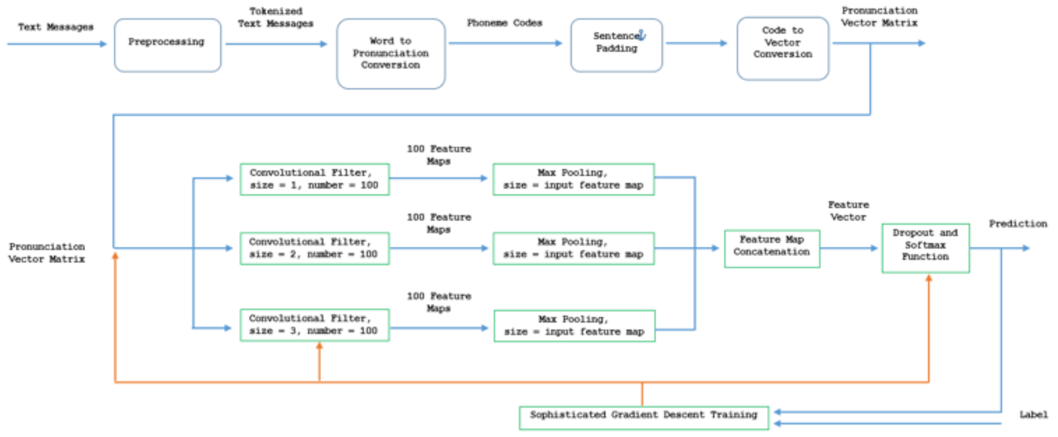

Text Messages
Preprocessing
Tokenized Text Messages
Word to Pronunciation Conversion
Phoneme Codes
Sentence Padding
Code to Vector Conversion
Pronunciation Vector Matrix
100 Feature Maps
Convolutional Filter, size = 1, number = 100
Max Pooling, size = input feature map
Convolutional Filter, size = 2, number = 100
Max Pooling, size = input feature map
Convolutional Filter, size = 3, number = 100
Max Pooling, size = input feature map
Pronunciation Vector Matrix
Feature Vector
Feature Map Concatenation
Dropout and Softmax Function
Prediction
Sophisticated Gradient Descent Training
Label

Figure 2: Architectural Design.

3 RESULTS AND DISCUSSION

3.1 Simulation

The convolutional neural networks (CNNs) AI/ML algorithm is considered the most widely utilized deep learning architecture in AI/ML image processing and recognition. This section discusses the importance of CNNs from a natural language processing perspective and a short Keras implementation with code explanations. Fig. 3 shows the idea of gliding or convolving pre-determined data is the central notion behind why CNNs are named the way they are.

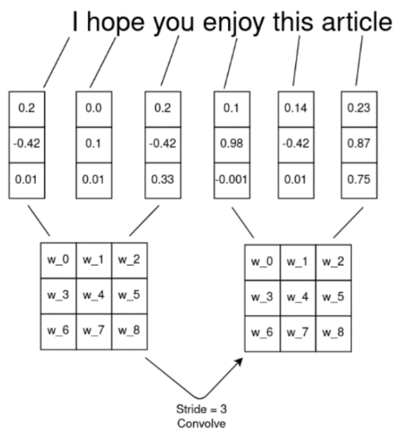


Figure 3: Convolution as described in Wikipedia.

The following essential phase is to calculate the convolved feature. This phase regard a 5×5 image and a 3×3 filter. Dealing with CNN's requires predominantly working with square matrices. Fig. 4 below displays how the first cell in the output layer is computed; the red numbers in the illustrated image designate the weights in the filter. As the individual filter slides over the data window one at a time, the output layer is computed by summing over the element-wise multiplication. Each pixel is multiplied by its corresponding weight in the filter.

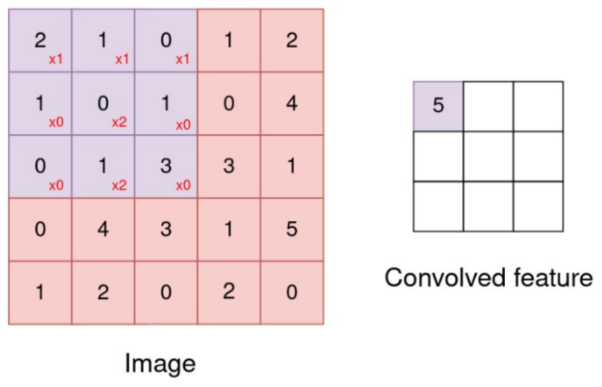

2 x1 | 1 x1 | 0 x1 | 1 | 2
1 x0 | 0 x2 | 1 x0 | 0 | 4
0 x0 | 1 x2 | 3 x0 | 3 | 1
0 | 4 | 3 | 1 | 5
1 | 2 | 0 | 2 | 0
Image
5
Convolved feature

Figure 4: Convolved Feature.

The calculation is as follows:

$(1\times2)+(1\times1)+(1\times0)+(0\times1)+(2\times0)+(0\times1)+(0\times0)+(2\times1)+(0\times4)= 5$

In the case of a 2D filter the size of the output layer can be calculated using the following formula:

$(N-F)/S +1$

where N = size of image , F = size of filter S = stride(1 in our case).

3.2 Keras Implementation of the Study

This study executed CNN for NLP sexual predation detection considering little in-depth analysis on the details of data pre-processing. Thus, the study proceeded with data vectorization and tokenization. Moreover, Fig. 5 shows a word2vec embedding was used in the provided example, with each token being characterized as a 300-Dimension word vector.

Figure 5: Word2Vec Embedding

```
word2vec_path = 'GoogleNews-vectors-negative300.bin.gz'
word2vec =
models.KeyedVectors.load_word2vec_format(word2vec_path,
binary=True)

def get_average_word2vec(tokens_list, vector,
generate_missing=False, k=300):
    if len(tokens_list)<1:
        return np.zeros(k)
    if generate_missing:
        vectorized = [vector[word] if word in vector else
np.random.rand(k) for word in tokens_list]
    else:
        vectorized = [vector[word] if word in vector else
np.zeros(k) for word in tokens_list]
    length = len(vectorized)
    summed = np.sum(vectorized, axis=0)
    averaged = np.divide(summed, length)
    return averaged

def get_word2vec_embeddings(vectors, clean_comments,
generate_missing=False):
    embeddings = clean_comments['tokens'].apply(lambda x:
get_average_word2vec(x, vectors,

generate_missing=generate_missing))
    return list(embeddings)

training_embeddings = get_word2vec_embeddings(word2vec,
data_train, generate_missing=True)

MAX_SEQUENCE_LENGTH = 50
EMBEDDING_DIM = 300
```

Figure 5: Word2Vec Embedding.

Fig. 6 shows that the accepted data was also padded so that individual sentences possessed 400 tokens. Lengthy sentences were trimmed off after 400 tokens, and more concise sentences were zero-padded. The consequential dimension for each sentence is 300×400.

Figure 6: Tokenize and Pad Sequences

```
tokenizer = Tokenizer(num_words=len(TRAINING_VOCAB),
lower=True, char_level=False)
tokenizer.fit_on_texts(data_train["Text_Final"].tolist())
training_sequences =
tokenizer.texts_to_sequences(data_train["Text_Final"].tolist())

train_word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(train_word_index))

Found 2638 unique tokens

train_cnn_data = pad_sequences(training_sequences,
maxlen=MAX_SEQUENCE_LENGTH)

train_embedding_weights = np.zeros((len(train_word_index)+1,
EMBEDDING_DIM))
for word,index in train_word_index.items():
    train_embedding_weights[index,:] = word2vec[word] if word
in word2vec else np.random.rand(EMBEDDING_DIM)
print(train_embedding_weights.shape)

(2639, 300)

test_sequences =
tokenizer.texts_to_sequences(data_test["Text_Final"].tolist())
test_cnn_data = pad_sequences(test_sequences,
maxlen=MAX_SEQUENCE_LENGTH)
```

Figure 6: Tokenize and Pad Sequences.

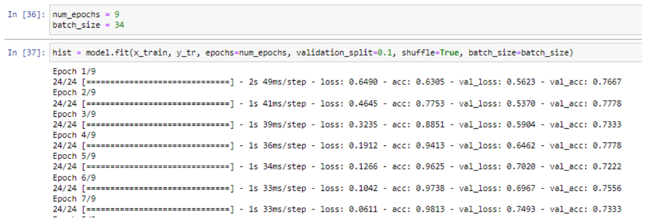
Fig. 7 shows that the hyperparameters were defined, and a CNN model was built.

Figure 7: Hyperparameter

```
def ConvNet(embeddings, max_sequence_length, num_words,
embedding_dim, labels_index):

  embedding_layer = Embedding(num_words,
                          embedding_dim,
                          weights=[embeddings],
                          input_length=max_sequence_length,
                          trainable=False)

    sequence_input = Input(shape=(max_sequence_length,),
dtype='int32')
    embedded_sequences = embedding_layer(sequence_input)

    convs = []
    filter_sizes = [2,3,4,5,6]

    for filter_size in filter_sizes:
        l_conv = Conv1D(filters=200, kernel_size=filter_size,
activation='relu')(embedded_sequences)
        l_pool = GlobalMaxPooling1D()(l_conv)
        convs.append(l_pool)

    l_merge = concatenate(convs, axis=1)

    x = Dropout(0.1)(l_merge)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.2)(x)
    preds = Dense(labels_index, activation='sigmoid')(x)

    model = Model(sequence_input, preds)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['acc'])
    model.summary()
    return model
```

Figure 7: Hyperparameter.

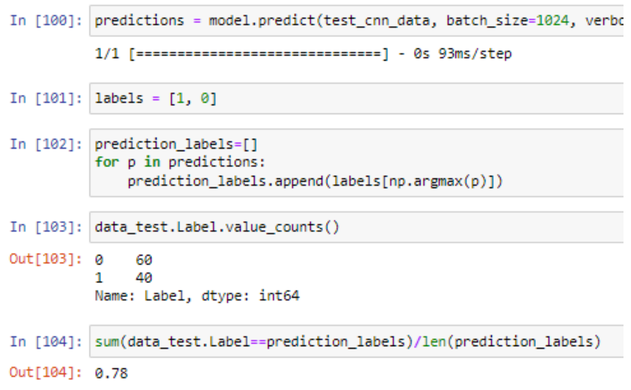
3.3 Predicting Sexual Predation with Keras CNN



```
In [36]: num_epochs = 9
         batch_size = 34

In [37]: hist = model.fit(x_train, y_tr, epochs=num_epochs, validation_split=0.1, shuffle=True, batch_size=batch_size)

         Epoch 1/9
         24/24 [==============================] - 2s 49ms/step - loss: 0.6490 - acc: 0.6305 - val_loss: 0.5623 - val_acc: 0.7667
         Epoch 2/9
         24/24 [==============================] - 1s 41ms/step - loss: 0.4645 - acc: 0.7753 - val_loss: 0.5370 - val_acc: 0.7778
         Epoch 3/9
         24/24 [==============================] - 1s 39ms/step - loss: 0.3235 - acc: 0.8851 - val_loss: 0.5904 - val_acc: 0.7333
         Epoch 4/9
         24/24 [==============================] - 1s 36ms/step - loss: 0.1912 - acc: 0.9413 - val_loss: 0.6462 - val_acc: 0.7778
         Epoch 5/9
         24/24 [==============================] - 1s 34ms/step - loss: 0.1266 - acc: 0.9625 - val_loss: 0.7020 - val_acc: 0.7222
         Epoch 6/9
         24/24 [==============================] - 1s 33ms/step - loss: 0.1042 - acc: 0.9738 - val_loss: 0.6967 - val_acc: 0.7556
         Epoch 7/9
         24/24 [==============================] - 1s 33ms/step - loss: 0.0611 - acc: 0.9813 - val_loss: 0.7493 - val_acc: 0.7333
```

Figure 8: Training set CNN.

To start with, a simple CNN was initially implemented to load the datasets. The CNN model developed has a relatively simple architecture: 4 convolutional layers, followed by a single densely-connected layer as shown in Fig. 8.



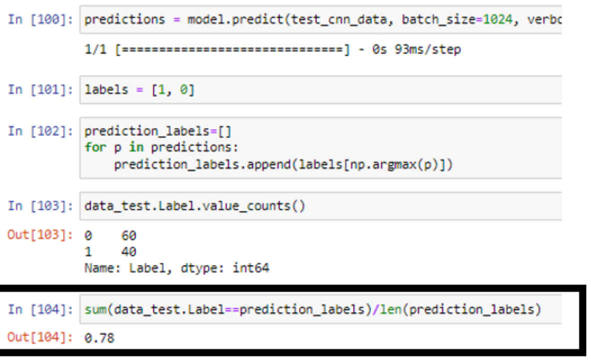
```
In [100]: predictions = model.predict(test_cnn_data, batch_size=1024, verbo

          1/1 [==============================] - 0s 93ms/step

In [101]: labels = [1, 0]

In [102]: prediction_labels=[]
          for p in predictions:
              prediction_labels.append(labels[np.argmax(p)])

In [103]: data_test.Label.value_counts()

Out[103]: 0    60
          1    40
          Name: Label, dtype: int64

In [104]: sum(data_test.Label==prediction_labels)/len(prediction_labels)

Out[104]: 0.78
```

Figure 9: Testing set CNN.

Fig. 9 shows the process performed by the proponent in training the model on the training set. At this stage, validation can also be set. After training phase, the next step is evaluating the model on the testing set.

Fig. 10 shows that the developed model achieves an accuracy of 78%, which is the baseline accuracy level for CNN NLP. Different results may be obtained because of the way weights are initialized. magnetic moment along the chain and field direction.

During the experimentation phase, a number of discrepancies from the output has been observed. As shown in Fig. 11, there were fifty-five (55) number of True Positive, meaning these are statements identified as predatory in nature. While five (5) fall under the False Positive category, showing that these statements were identified as predatory by the model, but in reality, they are not. On the other hand, there were seventeen (17) False Negative statement which means that these are statements that are predatory in nature but the model does not identify it as predatory. While twenty-three (23) fall under True Negative, which means that these statements are not predatory in nature and the model identified it correctly as non-predatory.



```
In [100]: predictions = model.predict(test_cnn_data, batch_size=1024, verbo
          1/1 [==============================] - 0s 93ms/step

In [101]: labels = [1, 0]

In [102]: prediction_labels=[]
          for p in predictions:
              prediction_labels.append(labels[np.argmax(p)])

In [103]: data_test.Label.value_counts()
Out[103]: 0    60
          1    40
          Name: Label, dtype: int64

In [104]: sum(data_test.Label==prediction_labels)/len(prediction_labels)
Out[104]: 0.78
```

Figure 10: Model Accuracy.

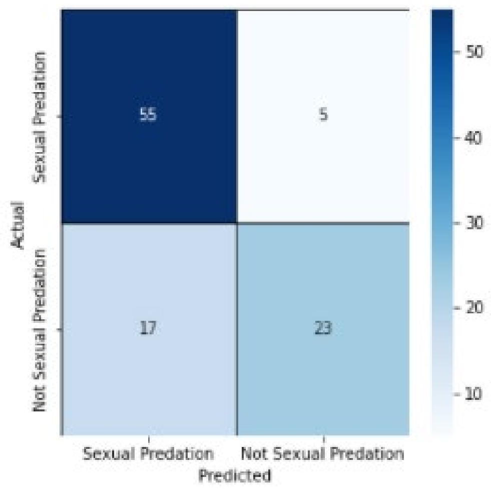
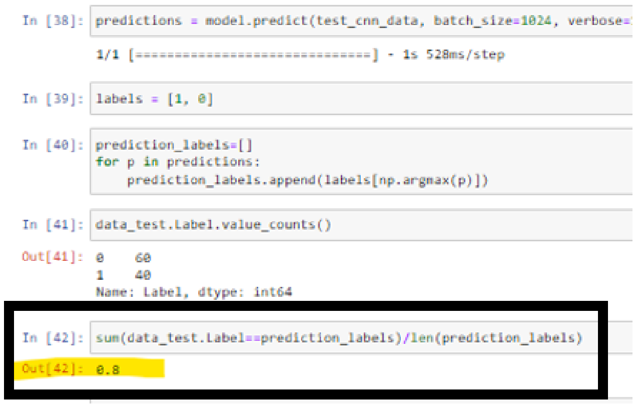


Figure 11: Result Summary of the Model.

3.4 Improving the Accuracy of the Model through Parameter Tuning

Parameter tuning is when a model learns the detail and noise in the training data to the extent that it negatively impacts the model's performance on new data. This indicates that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. Parameter tuning leads to an AI/ML model that models the training data too well. Fig. 12 shows that through parameter tuning, the model's accuracy was greatly improved from 78% which is a baseline accuracy score for CNN NLP AI/ML model up to 80%.



```
In [38]: predictions = model.predict(test_cnn_data, batch_size=1024, verbose=
         1/1 [==============================] - 1s 528ms/step

In [39]: labels = [1, 0]

In [40]: prediction_labels=[]
         for p in predictions:
             prediction_labels.append(labels[np.argmax(p)])

In [41]: data_test.Label.value_counts()
Out[41]: 0    60
         1    40
         Name: Label, dtype: int64

In [42]: sum(data_test.Label==prediction_labels)/len(prediction_labels)
Out[42]: 0.8
```

Figure 12: Parameter Tuning.

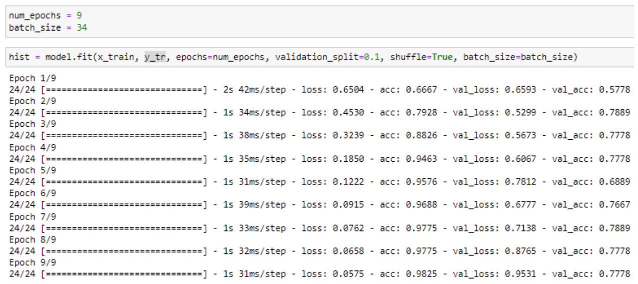
3.4 Evaluation Results

Big O Notation. When designing and using algorithms, a common question is which algorithm is more efficient at accomplishing a specific task. There are multiple strategies for determining efficiency, one of it is the Big-O. Big-O is concerned with an algorithm's asymptotic time complexity or the upper bound for how many steps it will take compared to how much data is processed. Formally, big-O is defined as:

$$f(n)=O(g(n)) \Longrightarrow \exists k\in R, k>0 \wedge \exists n0 \in R, n0 \geq 0 s.t. \forall n \geq n0 \mid f(n) \leq k \cdot g(n) \quad (1)$$

The above equation shows that there exists two constants k, n0 on the real number line such that f(n) is at most as large as k · g(n) for all n that's at least as big as n0. In other words, if after a particular point, the number of actions it takes the algorithm to execute is proportional to another function g, it may infer then that the big-O of that algorithm can be expressed as g.

Fig. 13 below shows that the model has a Big O complexity which means that it is efficient in accomplishing the task particularly for the detection of cybersexual predation. As observed, the proponent performed an automated simulation using python for data science in order to test the efficiency of the algorithm. In computing, an epoch is a date and time from which a computer measures system time. An epoch stands for time, and after 9th epoch, our algorithm scored 0.9825 (×100)= 98.25%.



```
num_epochs = 9
batch_size = 34
```

```
hist = model.fit(x_train, y_tr, epochs=num_epochs, validation_split=0.1, shuffle=True, batch_size=batch_size)
```

```
Epoch 1/9
24/24 [==============================] - 2s 42ms/step - loss: 0.6504 - acc: 0.6667 - val_loss: 0.6593 - val_acc: 0.5778
Epoch 2/9
24/24 [==============================] - 1s 34ms/step - loss: 0.4530 - acc: 0.7928 - val_loss: 0.5299 - val_acc: 0.7889
Epoch 3/9
24/24 [==============================] - 1s 38ms/step - loss: 0.3239 - acc: 0.8826 - val_loss: 0.5673 - val_acc: 0.7778
Epoch 4/9
24/24 [==============================] - 1s 35ms/step - loss: 0.1850 - acc: 0.9463 - val_loss: 0.6067 - val_acc: 0.7778
Epoch 5/9
24/24 [==============================] - 1s 31ms/step - loss: 0.1222 - acc: 0.9576 - val_loss: 0.7812 - val_acc: 0.6889
Epoch 6/9
24/24 [==============================] - 1s 39ms/step - loss: 0.0915 - acc: 0.9688 - val_loss: 0.6777 - val_acc: 0.7667
Epoch 7/9
24/24 [==============================] - 1s 33ms/step - loss: 0.0762 - acc: 0.9775 - val_loss: 0.7138 - val_acc: 0.7889
Epoch 8/9
24/24 [==============================] - 1s 32ms/step - loss: 0.0658 - acc: 0.9775 - val_loss: 0.8765 - val_acc: 0.7778
Epoch 9/9
24/24 [==============================] - 1s 31ms/step - loss: 0.0575 - acc: 0.9825 - val_loss: 0.9531 - val_acc: 0.7778
```

Figure 13. Big O Notation of CNN

4 CONCLUSIONS

In this paper, the Convolutional Neural Network (CNN) architecture for the detection of cybersexual predation is studied. The CNN was utilized to visualize the features of the original statements from PAN12, perverted-justice.com datasets. Each of the kernels of the convolutional layers of our CNN model learned to extract messages which are predatory or not. Moreover, the tool was integrated with a CNN to extract online message transcripts and to classify whether messages are predatory or not (determination of its polarity). The dataset taken from PAN12, perverted-justice.com, was reviewed using the Exploratory Data Analysis and preprocessed using term compression to have a cleaner dataset. Then word-to-vector conversion using Google News Word2Vec converter was performed before it was processed in CNN Model as test and training datasets. Finally, the chosen model was tested for its accuracy and time complexity using Big O Notation. The researchers performed an analysis on the efficiency of the model by considering the number of resources such as the time and space. Using Python, and through simulated implementation, the researchers observed that the algorithm, after 9th epoch, achieved an accuracy rating of 0.9825 (x100) = 98.25% and by using a batch size of 34. As such, the model developed for this study was able achieved an accuracy rating of 80%.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C2014333).

REFERENCES

[1] Black, P. J., Wollis, M., Woodworth, M., & Hancock, J. T. (2015). A linguistic analysis of grooming strategies of online child sex offenders: Implications for our understanding of predatory sexual behavior in an increasingly computer-mediated world. Child Abuse & Neglect, 44, 140–149. doi:10.1016/j.chiabu.2014.12.004

[2] Kulsrud, H. B. (2019). Detection of cyber grooming during an online conversation (Master's thesis, NTNU).

[3] Suman, L. N. (2018). Cyber trauma: An overview. Indian Journal of Clinical Psychology, 45(1), 7-17.

[4] Escalante, H. J., Villatoro-Tello, E., Juárez, A., Montes, M., & Villaseñor-Pineda, L. (2013, June). Sexual predator detection in chats with chained classifiers. In Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (pp. 46-54).

[5] McGhee, I., Bayzick, J., Kontostathis, A., Edwards, L., McBride, A., & Jakubowski, E. (2011). Learning to Identify Internet Sexual Predation. International Journal of Electronic Commerce, 15(3), 103–122. doi:10.2753/jec1086-4415150305.

[6] Inches, G., & Crestani, F. (2012, September). Overview of the International Sexual Predator Identification Competition at PAN-2012. In CLEF (Online working notes/labs/workshop) (Vol. 30)

[7] Zhang, Xiang, Jonathan Tong, Nishant Vishwamitra, Elizabeth Whittaker, Joseph P. Mazer, Robin Kowalski, Hongxin Hu, Feng Luo, Jamie Macbeth, and Edward Dillon. "Cyberbullying detection with a pronunciation based convolutional neural network." In 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 740-745. IEEE, 2016.

[8] Ebrahimi, M., Suen, C. Y., & Ormandjieva, O. (2016). Detecting predatory conversations in social media by deep convolutional neural networks. Digital Investigation, 18, 33-49.