Concept Explanation

The concept is to build a deep learning model using a Convolutional Neural Network (CNN) to classify sentences as either containing SQL injection attempts or not. SQL injection is a common web security vulnerability that allows attackers to interfere with the queries that an application makes to its database. Detecting such attempts is crucial for maintaining the security of web applications.  
  
Table of Contents

1. Imports and Library Setup
2. Reading and Preprocessing Data
3. Building the CNN Model
4. Training the CNN Model
5. Evaluating the CNN Model
6. Visualization
7. References

Certainly! Here's a summarized explanation of the process:

Data Extraction and Overview

The dataset is loaded from a CSV file containing textual data and corresponding labels. The features (textual data) and labels are extracted from the dataset. A brief overview of the dataset's dimensions, including the number of samples and feature dimensionality, is provided.

Understanding Dataset Content

A subset of the dataset, typically the first few rows, is displayed to give a preview of the extracted features and labels. This allows users to inspect the format and content of the data, facilitating a better understanding of the dataset's structure and content

Text Data Preprocessing

To prepare the text data for modeling, a series of preprocessing steps are performed:

Importing NLTK: The Natural Language Toolkit (NLTK) library is imported to access various natural language processing tools and resources.

Downloading Stopwords: Stopwords are common words (e.g., "the", "is", "and") that are often removed from text data as they typically do not carry much meaning or contribute to the classification task. The NLTK stopwords corpus is downloaded to access a predefined list of stopwords. Model Prediction and Evaluation

After training the Convolutional Neural Network (CNN) model, it is used to predict the labels of the test set (X\_test1). The following steps are involved in this process:

Model Prediction:

The trained CNN model is used to predict the labels of the test set (X\_test1). This is done by calling the predict() method on the model object, passing the test data as input.

The predicted probabilities are obtained for each sample in the test set.

Thresholding Predictions:

Since the output of the model is probabilities, a thresholding step is performed to convert these probabilities into binary predictions. This is typically done by rounding the predicted probabilities to the nearest integer (0 or 1), using NumPy's np.round() function.

The thresholding step is necessary for binary classification tasks, where the model predicts the probability of the positive class (in this case, the presence of a SQL injection attack).

Model Evaluation:

The accuracy of the CNN model on the test set is calculated using the accuracy\_score() function from scikit-learn. Accuracy measures the proportion of correctly predicted instances out of all instances in the test set.

Additionally, the F1 score of the CNN model on the test set is calculated using the f1\_score() function. The F1 score is the harmonic mean of precision and recall and provides a balance between these two metrics.

Both accuracy and F1 score are commonly used evaluation metrics for binary classification tasks and provide insights into the model's overall performance.

Vectorization with CountVectorizer: The CountVectorizer class from the scikit-learn library is used to convert the text data into numerical feature vectors. This process involves tokenizing the text data, converting it into a matrix of token counts, and removing stopwords. The parameters min\_df and max\_df are used to specify the minimum and maximum document frequency thresholds for including terms in the vocabulary.

Splitting the Dataset: The preprocessed text data (X) and corresponding labels (y) are split into training and testing sets using the train\_test\_split function from scikit-learn. This allows for evaluating the model's performance on unseen data. The test\_size parameter determines the proportion of the dataset that will be used for testing.

Dataset Dimensions: The shapes of the training and testing sets (X\_train, X\_test, y\_train, y\_test) are printed to provide insights into the number of samples and feature dimensionality in each set.|

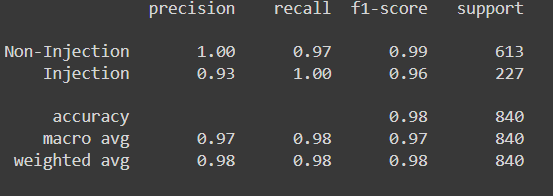
Building and Training the Convolutional Neural Network (CNN)

plot\_model(model, show\_shapes=True, to\_file='CNN.png')

Model Architecture:  
A screenshot of a computer

Description automatically generated

A Sequential model is initialized to build the CNN architecture.



Conclusion:

the CNN proves to be a robust ,faster and efficient approach for detecting SQL injection attempts in textual data. Its advantages over RNN, particularly in terms of efficiency, parallel processing, feature extraction, and translation invariance, make it a compelling choice for cybersecurity applications and beyond.