**Next Word Predictor**

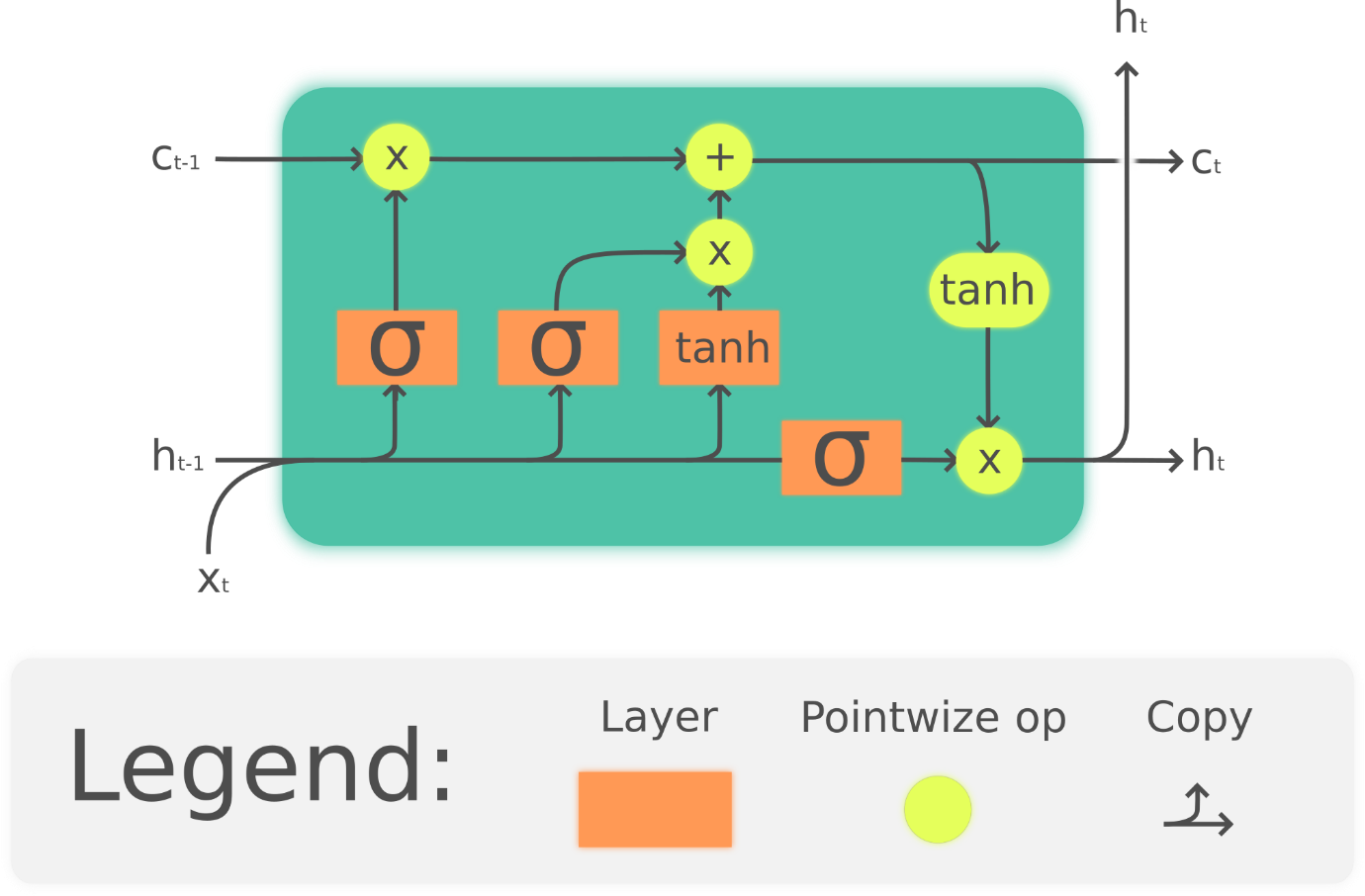
Aim: The objective of this project is to create **text prediction** model that predicts **words** using Natural Language Processing model (NLP) on a Deep Learning framework.

About The Model: Language prediction is a Natural Language Processing - NLP application concerned with predicting the text given in the preceding text. Auto-complete or suggested responses are popular types of language prediction. The first step towards language prediction is the selection of a language model. There are generally two models you can use to develop Next Word Suggester/Predictor: 1) N-grams model or 2) Long Short-Term Memory (LSTM). We will use LSTM as there are limitations with N-grams approach one of them being that we get suggestions based only on the frequency in N-grams approach hence, there are many scenarios where this approach could fail.

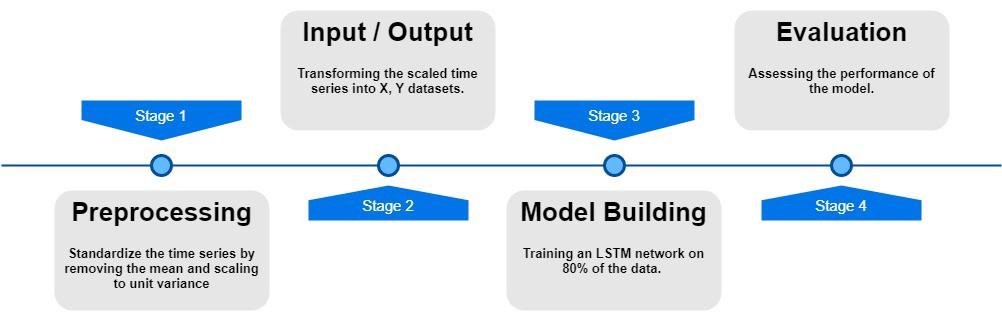
Dataset: We will use the text from the book ***Metamorphosis*** by Franz Kafka.

Frameworks: TensorFlow-Keras

Network Architecture: LSTM (Long Short-Term Memory)

******

Process Flow:



Data Pre-processing: The analysis of data set is done in several steps mentioned below.

1. **Loading the dataset:** Data loading is the process of copying and loading data or data sets from a source file, folder or application to a database or similar application. It is used in database-based extraction and loading techniques. Data can be loaded from several kind of files like csv, excel, pdf, word, sql, etc. The data when loaded is transformed into a data frame on which different operations are performed and model is trained.
2. **Data Cleaning:** Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results. There are several methods for cleaning data depending on how it is stored along with the answers being sought. Data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set’s accuracy without necessarily deleting information. The goal of data cleaning is to create data sets that are standardized and uniform to allow business intelligence and [data analytics tools](https://www.sisense.com/product/) to easily access and find the right data for each query. Following are few data cleaning operations that are to be executed on the given dataset-  
   * Replacing all the unnecessary extra new lines, the carriage return, and the Unicode character.
   * Making sure we have only unique words.
   * Considering each word only once and removing any additional repetitions.

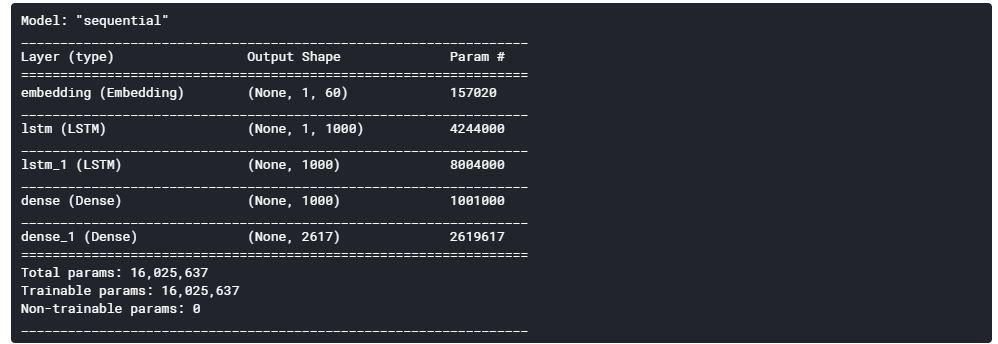
This will help the model train better avoiding extra confusion due to the repetition of words.

1. **Tokenization:** Tokenization refers to splitting bigger text data, essays, or corpora into smaller segments. These smaller segments can be in the form of smaller documents or lines of text data. They can also be a dictionary of words. We will then convert the texts to sequences. This is a way of interpreting the text data into numbers so that we can perform better analyses on them. We will then create the training dataset.
2. **Training Dataset:** The �X� will contain the training data with the input of text data. The �y� will contain the outputs for the training data. So, the �y� contains all the next word predictions for each input �X�. We will calculate the *vocab\_size* by using the length extracted from *tokenizer.word\_index* and then add 1 to it. We are adding 1 because 0 is a reserved for padding and we want to start our count from 1.
3. **Categorizing data:** Finally, we will convert our predictions data �y� to categorical data of the vocab size. This function converts a class vector (integers) to the binary class matrix. This will be useful with our loss which will be *categorical\_crossentropy*.
4. The rest of the code for the tokenization of data, creating the dataset, and converting the prediction set into categorical data

Data Modelling: We will be building a sequential model and proceed as follows-

* We will then create an embedding layer and specify the input dimensions and output dimensions.
* It is important to specify the input length as 1 since the prediction will be made on exactly one word and we will receive a response for that particular word.
* We will then add an LSTM layer to our architecture. We will give it 1000 units and make sure we return the sequences as true. This is to ensure that we can pass it through another LSTM layer.
* For the next LSTM layer, we will also pass it through another 1000 units but we don’t need to specify return sequence as it is false by default.
* We will pass this through a hidden layer with 1000 node units using the dense layer function with relu set as the activation.
* Finally, we pass it through an output layer with the specified vocab size and a softmax activation.
* The softmax activation ensures that we receive a bunch of probabilities for the outputs equal to the vocab size.
* After we look at the model code, we will also look at the model summary and the model plot.

#### Model Summary



#### Model Plot

#### 

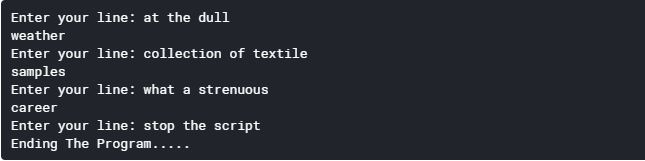
Compile and Fit:

* We are compiling and fitting our model in the final step. Here, we are training the model and saving the best weights to *nextword1.h5* so that we don’t have to re-train the model repeatedly and we can use our saved model when required.
* Here we have trained only on the training data. However, one can choose to train with both train and validation data.
* The loss we have used is *categorical\_crossentropy* which computes the cross-entropy loss between the labels and predictions.
* The optimizer we will be using is Adam with a learning rate of 0.001 and we will compile our model on the metric loss.

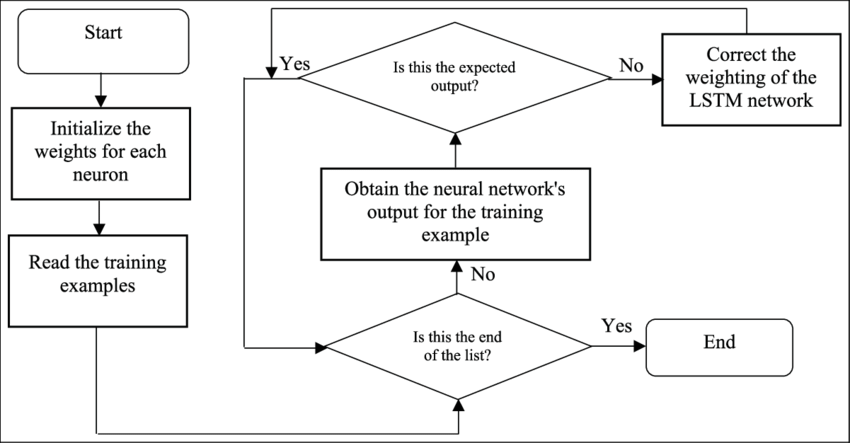


Prediction:

* For the prediction notebook, we will load the tokenizer file which we have stored in the pickle format.
* We will then load our next word model which we have saved in our directory.
* We will use this same tokenizer to perform tokenization on each of the input sentences for which we should make the predictions on.
* After this step, we can proceed to make predictions on the input sentence by using the saved model.
* We will use the try and except statements while running the predictions.
* We are using these statements because in case there is an error in finding the input sentence, we do not want the program to exit the loop.
* We want to run the script as long as the user wants the script to be run.
* When the user wants to exit the script, the user must manually choose to do so.
* The program will run as long as the user desires.



Flowchart :

******