Disclaimer: The following content is based on existing documentation on the topics covered as well as the author's personal experience.

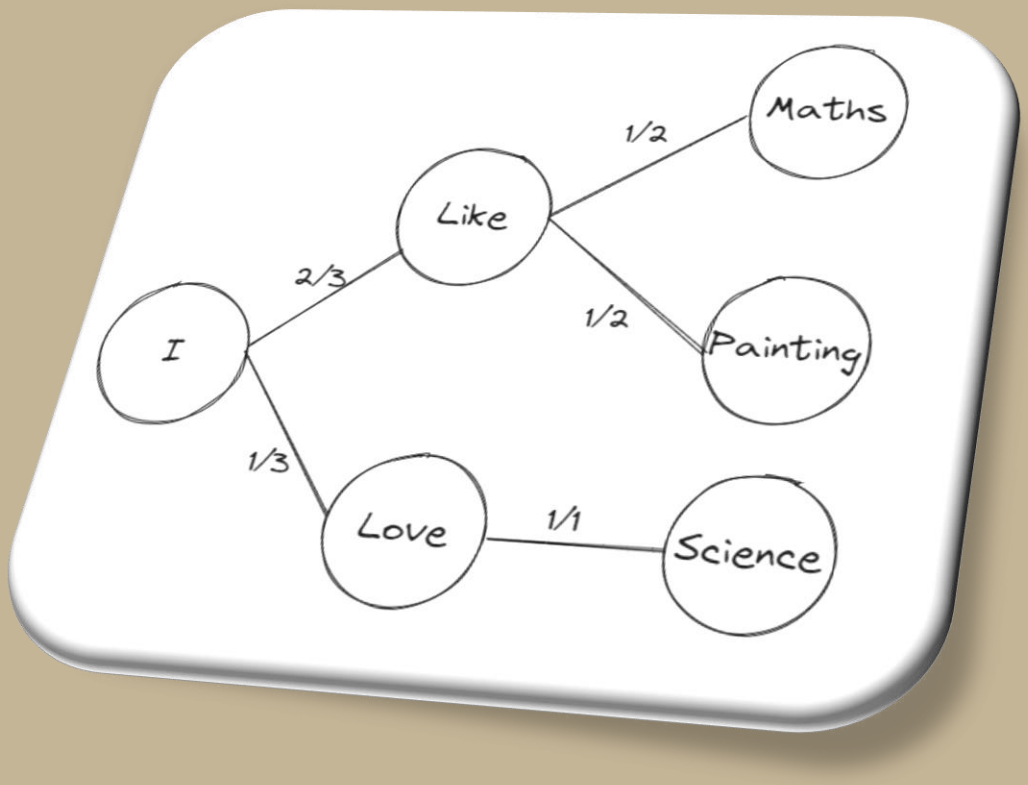
Written by Jhoeliel Palma Salazar | LinkedIn – Data Scientist

A Simple Next-Word Prediction Model

Next-word prediction has become an integral part of various applications ranging from text recommendation, speech recognition to chatbots, and virtual assistants. The main objective of these models is to generate human-like text by predicting the likelihood of the next word based on previous words. This paper presents a simple and fundamental approach to build a next-word prediction model using Python. The model is designed using Keras, a Python deep-learning library. We also discuss the limitations of the presented model and suggest potential improvements for future work.

Introduction:

The field of Natural Language Processing (NLP) has witnessed remarkable advancements with the introduction of deep learning techniques. One such application of deep learning in NLP is the prediction of the next word in a sequence, given the sequence of previous words. Despite its apparent simplicity, this task is challenging due to the complex structure of natural languages.



Methodology:

We employ a relatively straightforward approach in this study, where we use a sequence of words to predict the next word. Our sample text is a small excerpt from a popular literary piece. We tokenize the text, converting the words into integers, which allows us to process them in our model.

Our model consists of three layers. The first is an Embedding layer, which converts our integer-encoded vocabulary into dense vectors of fixed size. This is followed by a Long Short-Term Memory (LSTM) layer with 50 neurons. LSTM is a type of Recurrent Neural Network (RNN) well-suited for sequence prediction problems. The final layer is a Dense layer with a size equal to the vocabulary. It uses a softmax activation function to output a probability distribution for the next word.

Written by Jhoeliel Palma Salazar | LinkedIn – Data Scientist

```
model = Sequential()
model.add(Embedding(vocab_size, 10, input_length=1))
model.add(LSTM(50))
model.add(Dense(vocab_size, activation='softmax'))
```

After defining the model, we compile it using categorical cross-entropy as the loss function and Adam as the optimizer. The model is then trained for a pre-specified number of epochs.

Hands on:

This a way to implement this model using Python and its necessary libraries:

```
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences

# This is our sample sequence.
text = """In a place of La Mancha, whose name I do not want to remember, not long ago lived a hidalgo of
the kind with spear in shipyard, ancient shield, lean nag, and greyhound runner."""

# We tokenize the text to convert words into integers
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])
encoded = tokenizer.texts_to_sequences([text])[0]

# We determine the vocabulary size
vocab_size = len(tokenizer.word_index) + 1

# We create sequences of words -> word
sequences = list()
for i in range(1, len(encoded)):
    sequence = encoded[i-1:i+1]
    sequences.append(sequence)
sequences = np.array(sequences)

# We split into inputs and outputs
X, y = sequences[:,0],sequences[:,1]
y = to_categorical(y, num_classes=vocab_size)

# We define the model
model = Sequential()
model.add(Embedding(vocab_size, 10, input_length=1))
model.add(LSTM(50))
model.add(Dense(vocab_size, activation='softmax'))
print(model.summary())

# We compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# We train the model
model.fit(X, y, epochs=500, verbose=2)

# Function to predict the next word based on input
def predict_next_word(model, tokenizer, text):
    encoded = tokenizer.texts_to_sequences([text])[0]
    encoded = np.array(encoded)
    yhat = model.predict_classes(encoded)
    for word, index in tokenizer.word_index.items():
        if index == yhat:
            return word
    return None

# We test the model
print(predict_next_word(model, tokenizer, 'place'))
```

Written by Jhoeliel Palma Salazar | LinkedIn – Data Scientist

Note: In this example, we are using a very simple model and a small amount of text, so the predictions will not be accurate. In a real project, you would want to use a much larger dataset, fine-tune the model, and consider other techniques, like transformer-based language modeling, which is a more advanced and accurate technique for this type of tasks.

Results and Discussion

The model was able to predict the next word in the sequence with a certain degree of accuracy. However, the results are not very accurate due to the simplicity of the model and the small size of the training data.

Conclusion

This study provides a simple and foundational approach to next word prediction using Python and Keras. However, it should be noted that this is a preliminary study and the model used here is far from perfect. In practical applications, you'd want to use larger and more diverse datasets, fine-tune the model, and consider more advanced techniques, such as transformer-based models like GPT-3 and BERT, which have been shown to deliver superior performance on this type of task.

Despite the limitations, this work provides an excellent starting point for those interested in exploring the field of next word prediction using deep learning techniques. Further improvements can be made by expanding the training dataset, optimizing the model's parameters, and employing advanced models.