import pandas as pd

import numpy as np

import tensorflow as tf

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense, Bidirectional

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

# Load your dataset

data = pd.read\_excel(r"C:\Users\mrng shift\Desktop\New\_flavour\_data.xlsx")

# Tokenize the taste descriptions

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(data['Taste Description'])

vocab\_size = len(tokenizer.word\_index) + 1

# Encode the target descriptions (y values)

label\_encoder = LabelEncoder()

y\_encoded = label\_encoder.fit\_transform(data['Taste Description'])

# Prepare sequences of molecule combinations

max\_sequence\_length = 5 # Set an appropriate maximum sequence length

X = data[[' Molecules A', 'Molecules B']].values

X\_encoded = [tokenizer.texts\_to\_sequences([f"{m1}, {m2}"])[0] for m1, m2 in X]

X\_padded = pad\_sequences(X\_encoded, maxlen=max\_sequence\_length)

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_padded, y\_encoded, test\_size=0.2, random\_state=42)

# Define a Bidirectional LSTM model

model = Sequential()

model.add(Embedding(input\_dim=vocab\_size, output\_dim=47, input\_length=max\_sequence\_length))

model.add(Bidirectional(LSTM(47))) # Increased complexity

model.add(Dense(vocab\_size, activation='sigmoid')) # Output dimension matches vocab\_size

# Compile the model with categorical cross-entropy loss

model.compile(loss="categorical\_crossentropy", optimizer='RMSprop', metrics=['accuracy'])

model.summary()

# Train the model with more epochs

model.fit(X\_train, tf.keras.utils.to\_categorical(y\_train, num\_classes=vocab\_size), epochs=2, batch\_size=64, validation\_data=(X\_test, tf.keras.utils.to\_categorical(y\_test, num\_classes=vocab\_size)))

# Predict taste descriptions

def predict\_taste(molecule1, molecule2):

sequence = tokenizer.texts\_to\_sequences([f"{molecule1}, {molecule2}"])[0]

padded\_sequence = pad\_sequences([sequence], maxlen=max\_sequence\_length)

predicted\_sequence = model.predict(padded\_sequence)

predicted\_word\_index = np.argmax(predicted\_sequence)

predicted\_word = label\_encoder.inverse\_transform([predicted\_word\_index])[0]

return predicted\_word

# Example usage

molecule1 = 'Terpineol'

molecule2 = 'Borneol'

predicted\_taste = predict\_taste(molecule1, molecule2)

print(f"Predicted Taste Description: {predicted\_taste}")