# ass-5

November 16, 2023

## 1 one_hot_enocding

[ ]:
```
import pandas as pd
df = pd.DataFrame({"col1": ["red", "Sun", "Moon", "Earth", "Moon", "Venus"]})
print("The original data\n")
print(df)
print("____________________________________________________________________\n")
df_new = pd.get_dummies(df, columns=["col1"], prefix="Planet")
print(f"The transform data using get_dummies \n\n {df_new}")
```

## 2 Word To Vector

[ ]:
```
from gensim.models import Word2Vec
import nltk
from nltk.corpus import brown
```

[ ]:
```
nltk.download('brown')
data = brown.sents()  # Use the Brown corpus from NLTK as sample data
```

[ ]:
```
model = Word2Vec(data, min_count=1,  window=5)    # CBOW model using the gensim␣
 ↪library's Word2Vec
model.train(data, total_examples=len(data), epochs=5) # Model is trained on the␣
 ↪data with a specified number of epochs
print(data)
word_vectors = model.wv
similarity = word_vectors.similarity('woman', 'man')
print(f"Similarity between 'woman' and 'man': {similarity}")
```

## 3 NLP

[ ]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
import numpy as np
import regex as re
```

[ ]:
```
def file_to_sentence_list(file_path):
        with open(file_path, 'r') as file:
                text = file.read()
        sentences = [sentence.strip() for sentence in re.split(r'(?<=[.!?
 ↪])\s+', text) if sentence.strip()]
        return sentences

file_path = r'C:\Users\DELL\Desktop\DL\DL\PR 5 DL\Word_Predication\pizza.txt'
text_data = file_to_sentence_list(file_path)

# Tokenize the text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(text_data)
total_words = len(tokenizer.word_index) + 1

# Create input sequences
input_sequences = []
for line in text_data:
        token_list = tokenizer.texts_to_sequences([line])[0]
        for i in range(1, len(token_list)):
                n_gram_sequence = token_list[:i+1]
                input_sequences.append(n_gram_sequence)

# Pad sequences and split into predictors and label
max_sequence_len = max([len(seq) for seq in input_sequences])
input_sequences = np.array(pad_sequences(
        input_sequences, maxlen=max_sequence_len, padding='pre'))
X, y = input_sequences[:, :-1], input_sequences[:, -1]

# Convert target data to one-hot encoding
y = tf.keras.utils.to_categorical(y, num_classes=total_words)
```

[ ]:
```
# Define the model
model = Sequential()
model.add(Embedding(total_words, 10,input_length=max_sequence_len-1))
model.add(LSTM(128))
model.add(Dense(total_words, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',␣
 ↪metrics=['accuracy'])
# Train the model
model.fit(X, y, epochs=150, verbose=1)
```

[ ]:
```python
# Generate next word predictions
seed_text = "Pizza"
next_words = 5

for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences(
                [token_list], maxlen=max_sequence_len-1, padding='pre')
        predicted_probs = model.predict(token_list)
        predicted_word = tokenizer.index_word[np.argmax(predicted_probs)]
        seed_text += " " + predicted_word

print("Next predicted words:", seed_text)
```