0907753 Natural Languages Processing (Spring 2024)

Midterm Exam

الاسم:	KEY	رقم التسجيل:	

==========

Exam Instructions:

Duration: 90 minutes

Materials Allowed: Open book and notes. No electronic devices permitted.

Instructions: Please answer all three problems in the spaces provided. Each problem is worth 10 marks. Ensure clarity and conciseness in your answers.

==========

P1. You are given the following text sequence:

```
text = "Today’s NLP technologies are increasingly sophisticated and accessible. They're transforming how we interact with technology on a “daily” basis. Isn’t it wonderful?"
```

- Write a Python function named `preprocess_text` that takes this text as input and returns a list of words after applying the following preprocessing steps:
  1. Convert the text to lowercase.
  2. Replace smart quotes with standard versions (*e.g.*, to `'` and `"`).
  3. Remove all punctuation.
  4. Remove stopwords.
- You can use the `spacy` library for punctuation and stopwords removal.
- Example Output:
  - `today nlp technologies increasingly sophisticated …`

Solution:

```python
import spacy

nlp = spacy.load("en_core_web_sm")

def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Replace smart quotes
    replace_chars = {
        "’": "'",  # Right single quotation mark
        "‘": "'",  # Left single quotation mark
        "“": '"',  # Left double quotation mark
        "”": '"'   # Right double quotation mark
    }

    for char, replace_with in replace_chars.items():
        text = text.replace(char, replace_with)

    # Tokenize text
    doc = nlp(text)

    # Remove all punctuation
    filtered_doc = [token for token in doc if not token.is_punct]

    # Remove stopwords
    filtered_tokens = [token.text for token in filtered_doc
```

```
                        if not token.is_stop]

    filtered_text = " ".join(filtered_tokens)
    return filtered_text

# Example usage
text = "Today’s NLP technologies are increasingly sophisticated and
accessible. They're transforming how we interact with technology on a
daily basis. Isn’t it wonderful?"
print(preprocess_text(text))
```

P2. You are provided with two text sequences:

```
Text1 = "Machine learning provides systems the ability to automatically learn and improve from experience."
Text2 = "Artificial intelligence enables computers to understand complex data and make decisions."
```

- Write a Python function named `calculate_similarity` that computes the cosine similarity between these two texts after transforming them into TF-IDF vectors and reducing their dimensionality using PCA to two topics each.
- You can use the `TfidfVectorizer` from scikit-learn's `sklearn.feature_extraction.text` to generate TF-IDF vectors and `PCA` from `sklearn.decomposition` module to perform dimensionality reduction. Use `cosine_similarity` from `sklearn.metrics.pairwise` to compute the similarity.
- Expected output: Single floating-point number representing the cosine similarity between the reduced vectors of the two texts.
- When applying PCA, fit the model on the TF-IDF matrix of both texts to capture the variance across both before transforming them.

Solution:

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
from sklearn.metrics.pairwise import cosine_similarity

def calculate_similarity(text1, text2):
    # Initialize the TF-IDF Vectorizer
    vectorizer = TfidfVectorizer()

    # Fit and transform the texts into TF-IDF vectors
    tfidf_matrix = vectorizer.fit_transform([text1, text2])

    # Initialize PCA and reduce the dimensionality to 2 components
    pca = PCA(n_components=2)
    reduced_tfidf_matrix = pca.fit_transform(tfidf_matrix.toarray())

    # Compute the cosine similarity between the two reduced vectors
    # Since we have two vectors, cosine_similarity returns a 2x2 matrix,
    # we are interested in the similarity between text1 and text2
    similarity = cosine_similarity(reduced_tfidf_matrix)

    return similarity[0, 1]  # return the similarity of 1st & 2nd text

# Example texts
text1 = "Machine learning provides systems the ability to automatically learn and improve from experience."
text2 = "Artificial intelligence enables computers to understand complex data and make decisions."

# Calculate the similarity
similarity_score = calculate_similarity(text1, text2)
print("Cosine Similarity:", similarity_score)
```

P3. The following Python code loads and preprocesses a labeled text dataset. This dataset is provided in the file named `training_data.csv`, which has two columns: `text` (the text sequence) and `label` (the category label). The preprocessing steps include preparing the labels, tokenization, and padding.

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

# Load the data
data = pd.read_csv('training_data.csv')
texts = data['text'].values
labels = data['label'].values

# Convert labels to categorical
labels = to_categorical(labels, num_classes=4)

# Tokenize text
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)
vocab_size = len(tokenizer.word_index) + 1
sequences = tokenizer.texts_to_sequences(texts)

# Find the maximum length of any text in the dataset
max_length = max(len(s) for s in sequences)

# Pad sequences to ensure uniform length
sequences_padded = pad_sequences(sequences, maxlen=max_length)
```

- Complete this code to build a Recurrent Neural Network (RNN) using Keras to classify text sequences into one of four categories. Your RNN must have the following architecture:
  1. An embedding layer of dimensionality 100
  2. Two LSTM layers each with 128 cells
  3. An output layer

Solution:

```
# Build the RNN model architecture
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=100,
              input_length=max_length, mask_zero=True),
    LSTM(128, return_sequences=True),
    LSTM(128),
    Dense(4, activation='softmax')
])
```

<Good Luck>