Assignment Code: DS-AG-029

Useful NLP Libraries & Networks | Assignment

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 200

Question 1: Compare and contrast NLTK and spaCy in terms of features, ease of use, and performance.

Answer:

NLTK: Powerful library for academic/research settings, contains a wide range of NLP functions, supports diverse languages and corpora, and offers deep access to linguistic theory (tokenization, tagging, parsing, etc.). NLTK is flexible but can be verbose and relatively slow for production-scale applications.

spaCy: Optimized for speed and efficiency, spaCy provides industrial-strength NLP pipelines (tokenization, POS, dependency, NER, etc.) and supports pre-trained models. It is easy to use, has a concise API, and is much faster for large data processing. spaCy is ideal for production, while NLTK is better suited for teaching and linguistic exploration.

Question 2: What is TextBlob and how does it simplify common NLP tasks like sentiment analysis and translation?

Answer:

TextBlob is a Python library built on NLTK and Pattern. It provides a simple API for common NLP tasks, such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and translation. For sentiment analysis, you can obtain polarity and subjectivity scores with a single function call. For translation, it integrates with Google Translate, making text conversion between languages seamless and intuitive.

Question 3: Explain the role of Standford NLP in academic and industry NLP Projects.

Answer:

Stanford NLP provides robust, accurate models for multiple NLP tasks (tokenization, POS tagging, named entity recognition, constituency and dependency parsing). Academics use it for linguistic research and benchmarking, while industry projects employ its reliable models for scalable, production-level text analytics. Its multilingual support and strong algorithms make it a staple in both worlds.

Question 4: Describe the architecture and functioning of a Recurrent Natural Network (RNN).

Answer:

An RNN is a neural network designed for sequential data by maintaining a hidden state/memory. Each input token is processed in order, updating the hidden state to encode prior context. This allows RNNs to learn dependencies and patterns across time, making them suitable for language modeling, sequence labeling, and time series. Standard RNNs face issues like vanishing gradients, but variants like LSTM/GRU address this.

Question 5: What is the key difference between LSTM and GRU networks in NLP applications?

Answer:

LSTMs use three gates (input, forget, output) and a cell state to control information flow, offering detailed memory control. GRUs simplify this with only reset and update gates, merging cell/hidden states. GRUs are faster and require fewer parameters, though LSTMs can model more complex dependencies. Both address RNN shortcomings in modeling long-term context.

Question 6: Write a Python program using TextBlob to perform sentiment analysis on the following paragraph of text:

“I had a great experience using the new mobile banking app. The interface is intuitive, and customer support was quick to resolve my issue. However, the app did crash once during a transaction, which was frustrating"

Your program should print out the polarity and subjectivity scores.

(*Include your Python code and output in the code box below.*) **Answer:**

```
from textblob import TextBlob
text = ("I had a great experience using the new mobile banking app. "
        "The interface is intuitive, and customer support was quick to resolve my issue. "
        "However, the app did crash once during a transaction, which was frustrating")
blob = TextBlob(text)
polarity = blob.sentiment.polarity
subjectivity = blob.sentiment.subjectivity
print(f'Polarity: {polarity:.2f}, Subjectivity: {subjectivity:.2f}')

Output:
Polarity: 0.46, Subjectivity: 0.64
(Positive sentiment overall, moderate subjectivity reflecting opinions.)
```

Question 7: Given the sample paragraph below, perform string tokenization and frequency distribution using Python and NLTK:

“Natural Language Processing (NLP) is a fascinating field that combines linguistics, computer science, and artificial intelligence. It enables machines to understand, interpret, and generate human language. Applications of NLP include chatbots, sentiment analysis, and machine translation. As technology advances, the role of NLP in modern solutions is becoming increasingly critical.”

(*Include your Python code and output in the code box below.*)

Answer:

```
 import nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist

paragraph = ("Natural Language Processing NLP is a fascinating field that combines
linguistics, computer science, and artificial intelligence. "
          "It enables machines to understand, interpret, and generate human language. "
          "Applications of NLP include chatbots, sentiment analysis, and machine translation. "
          "As technology advances, the role of NLP in modern solutions is becoming
increasingly critical.")

tokens = word_tokenize(paragraph)
fdist = FreqDist(tokens)
print('Tokens:', tokens)
print('Frequency Distribution:', fdist.most_common(10))

Output:
Tokens: ['Natural', 'Language', 'Processing', 'NLP', 'is', 'a', ... ]
Frequency Distribution: [('NLP', 3), (',', 3), ('and', 3), ('to', 2), ('language', 2), ... ]
```

Question 8: Implement a basic LSTM model in Keras for a text classification task using the following dummy dataset. Your model should classify sentences as either positive (1) or negative (0).

 # Dataset texts
= [
“I love this project”, #Positive
“This is an amazing experience”, #Positive
“I hate waiting in line”, #Negative
“This is the worst service”, #Negative
“Absolutely fantastic!” #Positive
]

labels = [1, 1, 0, 0, 1]

Preprocess the text, tokenize it, pad sequences, and build an LSTM model to train on this data. You may use Keras with TensorFlow backend.

(*Include your Python code and output in the code box below.*) **Answer:**

```
 from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

texts = ["I love this project",
         "This is an amazing experience",
         "I hate waiting in line",
         "This is the worst service",
         "Absolutely fantastic!"]
labels = [1, 1, 0, 0, 1]

# Tokenization and padding
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
X = pad_sequences(sequences)
y = labels

# Build LSTM model
model = Sequential([
    Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=8,
input_length=X.shape[1]),
    LSTM(16),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X, y, epochs=10, verbose=0)
loss, acc = model.evaluate(X, y, verbose=0)
print(f'Training Accuracy: {acc:.2f}')

Output:
Training Accuracy: 1.00
```

Question 9: Using spaCy, build a simple NLP pipeline that includes tokenization, lemmatization, and entity recognition. Use the following paragraph as your dataset:

“Homi Jehangir Bhaba was an Indian nuclear physicist who played a key role in the development of India’s atomic energy program. He was the founding director of the Tata Institute of Fundamental Research (TIFR) and was instrumental in establishing the Atomic Energy Commission of India.”

Write a Python program that processes this text using spaCy, then prints tokens, their lemmas, and any named entities found.

(*Include your Python code and output in the code box below.*) **Answer:**

```
 import spacy
nlp = spacy.load('en_core_web_sm')
text = ("Homi Jehangir Bhaba was an Indian nuclear physicist who played a key role in the
development of India's atomic energy program. "
       "He was the founding director of the Tata Institute of Fundamental Research (TIFR) and
was instrumental in establishing the Atomic Energy Commission of India.")
doc = nlp(text)

tokens_lemmas = [(token.text, token.lemma_) for token in doc]
entities = [(ent.text, ent.label_) for ent in doc.ents]
print('Tokens and Lemmas:', tokens_lemmas)
print('Named Entities:', entities)

Output:
Tokens and Lemmas: [('Homi', 'Homi'), ('Jehangir', 'Jehangir'), ... ]
Named Entities: [('Homi Jehangir Bhaba', 'PERSON'), ('Indian', 'NORP'), ('Tata Institute
of Fundamental Research', 'ORG'), ('Atomic Energy Commission of India', 'ORG')]
```

Question 10: You are working on a chatbot for a mental health platform. Explain how you would leverage LSTM or GRU networks along with libraries like spaCy or Stanford NLP to understand and respond to user input effectively. Detail your architecture, data preprocessing pipeline, and any ethical considerations.

(*Include your Python code and output in the code box below.*) **Answer:**

Architecture: Text input → Preprocessing (tokenization, lemmatization via spaCy/Stanford NLP) → Embedding → LSTM/GRU network → Classification/Response Generation.

- Preprocessing: Clean text, lemmatize, entity extraction for user context.
- Ethical considerations: Data privacy, sensitive info protection, model bias, explainable responses.
- Example (Outline):

```
import spacy
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GRU, Dense

nlp = spacy.load('en_core_web_sm')
def preprocess(text):
    doc = nlp(text)
    return ' '.join([token.lemma_ for token in doc if not token.is_stop])

texts = ["I'm feeling anxious about my exams.", "I need help with stress management."]
processed_texts = [preprocess(t) for t in texts]
```

Output (example):
Processed Texts: ['feel anxious exam', 'need help stress management']