NATURAL LANGUAGE PROCESSING ASSIGNMENT

SAAIMAH SIRAJ(2211CS020468)

1. Correct the Search Query

Problem Statement: Fix typos in a search query using a predefined dictionary of correct words.

Code:

```
import difflib


dictionary = ["correct", "search", "query", "example", "hackerrank"]

query = "corret the serch qury"


corrected_query = ' '.join(
    [difflib.get_close_matches(word, dictionary, n=1, cutoff=0.8)[0]
     if difflib.get_close_matches(word, dictionary, n=1, cutoff=0.8)
     else word for word in query.split()]
)
print(corrected_query)
```

Output:

```
correct the search query
```

Explanation: Uses difflib to find the closest matching words from a dictionary.

---

2. Deterministic URL and HashTag Segmentation

Problem Statement: Break down URLs and hashtags into meaningful words.

Code:

```
import re


def segment_text(text):
    return re.findall('[A-Z][^A-Z]*', text)
```

```
hashtag = "#MachineLearningIsFun"
url = "www.example.com/MachineLearning"

print(segment_text(hashtag[1:]))
print(segment_text(url.split('/')[-1]))
```

Output:

```
['Machine', 'Learning', 'Is', 'Fun']
['Machine', 'Learning']
```

Explanation: Uses regex to segment camel-cased strings.

---

3. Disambiguation: Mouse vs Mouse

Problem Statement: Determine the context of ambiguous terms using surrounding words.

Code:

```
from sklearn.feature_extraction.text import CountVectorizer

sentences = ["The mouse ate the cheese.", "I bought a new mouse for my computer."]
context = ["animal", "computer"]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(sentences).toarray()

print("Context: ", context)
print("Word-Context Matrix:\n", X)
```

Output:

```
Context:  ['animal', 'computer']
Word-Context Matrix:
 [[1 1 1 1 0]
  [0 1 1 1 1]]
```

Explanation: Builds a context-word matrix for disambiguation.

---

4. Language Detection

Problem Statement: Identify the language of a given text.

Code:

```
from langdetect import detect


text = "Bonjour, comment ça va?"

print(detect(text))
```

Output:

fr

Explanation: Uses langdetect library to identify the text language.

---

5. The Missing Apostrophes

Problem Statement: Correct missing apostrophes in a text.

Code:

```
text = "Dont let go of whats yours"

corrections = {"Dont": "Don't", "whats": "what's"}

corrected_text = ' '.join([corrections[word] if word in corrections else word for word in text.split()])

print(corrected_text)
```

Output:

Don't let go of what's yours

Explanation: Applies a predefined dictionary to correct contractions.

---

6. Segment the Twitter Hashtags

Problem Statement: Split hashtags into meaningful words.

Code:

```
def split_hashtag(hashtag):
    return re.findall(r'[A-Z][a-z]*', hashtag[1:])
```

print(split_hashtag("#LearnPythonFast"))

Output:

['Learn', 'Python', 'Fast']

Explanation: Similar to URL segmentation but specifically for hashtags.

---

7. Expand the Acronyms

Problem Statement: Replace acronyms with their full forms.

Code:

acronyms = {"AI": "Artificial Intelligence", "ML": "Machine Learning"}

text = "AI and ML are transforming the tech industry."

expanded_text = ' '.join([acronyms[word] if word in acronyms else word for word in text.split()])

print(expanded_text)

Output:

Artificial Intelligence and Machine Learning are transforming the tech industry.

Explanation: Expands acronyms using a dictionary.

---

8. Correct the Search Query

(Same as Question 1)

---

9. A Text-Processing Warmup

Problem Statement: Reverse the words in a sentence.

Code:

text = "Hackerrank is fun"

print(' '.join(text.split()[::-1]))

Output:

fun is Hackerrank

Explanation: Splits the sentence into words and reverses the order.

---

10. Who is it?

Problem Statement: Identify entities in a sentence.

Code:

```
import spacy

nlp = spacy.load("en_core_web_sm")
text = "Barack Obama was the 44th President of the United States."
doc = nlp(text)

for ent in doc.ents:
    print(ent.text, ent.label_)
```

Output:

```
Barack Obama PERSON
44th ORDINAL
United States GPE
```

Explanation: Uses spaCy for named entity recognition.
