```
Start coding or generate with AI.
```

Task

Perform text preprocessing on a user-provided medical/health text using NLTK for sentence tokenization, word tokenization, stemming (Porter Stemmer), and lemmatization (WordNetLemmatizer with POS tags), and spaCy for tokenization and lemmatization, then compare the outputs of stemming and lemmatization.

Install and Import Libraries

Subtask:

Install the `nltk` and `spaCy` libraries, then download NLTK data like `punkt` and `wordnet`. Import the required modules for tokenization, stemming, and lemmatization.

Reasoning: Install the required libraries, `nltk` and `spaCy`, using pip.

```
pip install nltk spacy
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packages (3.8.11)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nl
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/dist-packages (frc
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/dist-packages (frc
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/dist-packages (frc
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.12/dist-packages (from spac
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/d
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.1
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from spacy)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from sp
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from reque
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from ji
```

```
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open<8
```

Reasoning: Now that the libraries are installed, I will download the required NLTK data packages (`punkt` and `wordnet`) and then import all the necessary modules for tokenization, stemming, and lemmatization as specified in the instructions.

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')

from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
import spacy

print("NLTK data downloaded and modules imported.")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
NLTK data downloaded and modules imported.
```

Reasoning: I will initialize the NLTK WordNetLemmatizer and download the small English language model for spaCy, as spaCy requires a model for tokenization and lemmatization.

```
nltk_lemmatizer = WordNetLemmatizer()

# Download spaCy English model
try:
    spacy.load('en_core_web_sm')
except OSError:
    print('Downloading spaCy model en_core_web_sm...')
    spacy.cli.download('en_core_web_sm')

print("WordNetLemmatizer initialized and spaCy model checked/downloaded.")
```

```
WordNetLemmatizer initialized and spaCy model checked/downloaded.
```

Load Medical/Health Text

Subtask:

Define a variable to hold the medical/health text that will be used as the input corpus for preprocessing. The user will need to provide this text.

Reasoning: Define a variable to store the medical/health text, which will serve as the input for subsequent preprocessing steps, as instructed.

```
medical_text = """The patient presented with severe abdominal pain and nausea. A preliminary diag

print("Medical text defined.")
```

```
Medical text defined.
```

NLTK Sentence Tokenization

Subtask:

Apply NLTK's `sent_tokenize` function to split the loaded text into individual sentences.

Reasoning: I will apply NLTK's `sent_tokenize` function to the `medical_text` variable to split it into individual sentences, store them in `nltk_sentences`, and then print the result.

Reasoning: The previous code failed because the `punkt_tab` resource required by NLTK's `sent_tokenize` function was not downloaded. I will add `nltk.download('punkt_tab')` to ensure all necessary NLTK data is available before performing sentence tokenization, and then re-execute the sentence tokenization as instructed.

```
import nltk
nltk.download('punkt_tab') # Download 'punkt_tab' resource
nltk_sentences = sent_tokenize(medical_text)
print("NLTK Sentence Tokenization:")
for i, sentence in enumerate(nltk_sentences):
    print(f"Sentence {i+1}: {sentence}")
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
NLTK Sentence Tokenization:
Sentence 1: The patient presented with severe abdominal pain and nausea.
Sentence 2: A preliminary diagnosis of appendicitis was made based on clinical symptoms and elevat
Sentence 3: Further investigation, including a CT scan, is recommended to confirm the diagnosis an
Sentence 4: The patient has a history of gastroesophageal reflux disease (GERD) and occasional mig
```

Reasoning: Now that sentence tokenization is complete, the next logical step according to the task is to perform word tokenization using NLTK. I will iterate through the previously generated NLTK sentences and apply NLTK's `word_tokenize` function to each sentence to obtain word tokens.

```
nltk_word_tokens = []
print("NLTK Word Tokenization:")
for i, sentence in enumerate(nltk_sentences):
    words = word_tokenize(sentence)
    nltk_word_tokens.append(words)
    print(f"Sentence {i+1} Words: {words}")
```

```
NLTK Word Tokenization:
Sentence 1 Words: ['The', 'patient', 'presented', 'with', 'severe', 'abdominal', 'pain', 'and', 'n
Sentence 2 Words: ['A', 'preliminary', 'diagnosis', 'of', 'appendicitis', 'was', 'made', 'based',
Sentence 3 Words: ['Further', 'investigation', ',', 'including', 'a', 'CT', 'scan', ',', 'is', 're
Sentence 4 Words: ['The', 'patient', 'has', 'a', 'history', 'of', 'gastroesophageal', 'reflux', 'd
```

Reasoning: Now that word tokenization using NLTK is complete, the next step as per the task description is to perform stemming using NLTK's Porter Stemmer. I will initialize the PorterStemmer and then iterate through the `nltk_word_tokens` to apply stemming to each word.

```
porter_stemmer = PorterStemmer()
nltk_stemmed_tokens = []
print("NLTK Stemming (Porter Stemmer):")
for i, sentence_words in enumerate(nltk_word_tokens):
    stemmed_sentence = [porter_stemmer.stem(word) for word in sentence_words]
    nltk_stemmed_tokens.append(stemmed_sentence)
    print(f"Sentence {i+1} Stemmed: {stemmed_sentence}")
```

```
NLTK Stemming (Porter Stemmer):
Sentence 1 Stemmed: ['the', 'patient', 'present', 'with', 'sever', 'abdomin', 'pain', 'and', 'naus
Sentence 2 Stemmed: ['a', 'preliminari', 'diagnosi', 'of', 'append', 'wa', 'made', 'base', 'on', '
Sentence 3 Stemmed: ['further', 'investig', ',', 'includ', 'a', 'ct', 'scan', ',', 'is', 'recommen
Sentence 4 Stemmed: ['the', 'patient', 'ha', 'a', 'histori', 'of', 'gastroesophag', 'reflux', 'dis
```

Reasoning: Now that stemming is complete, the next step as per the task description is to perform lemmatization using NLTK's `WordNetLemmatizer`. For effective lemmatization, I need to convert NLTK's Part-of-Speech (POS) tags to WordNet's POS tags. I will define a helper function for this conversion, then iterate through the `nltk_word_tokens` and apply lemmatization to each word.

Reasoning: The previous code failed because the `averaged_perceptron_tagger_eng` resource, required by `nltk.pos_tag` for lemmatization with POS tags, was not downloaded. I will add `nltk.download('averaged_perceptron_tagger')` to ensure this resource is available before proceeding with lemmatization.

Reasoning: The previous code failed because it downloaded the generic `averaged_perceptron_tagger` instead of the specific `averaged_perceptron_tagger_eng` resource required by `nltk.pos_tag`. I will correct the download command to fetch `averaged_perceptron_tagger_eng` to resolve this `LookupError` and then re-execute the lemmatization step.

```
from nltk.corpus import wordnet
import nltk # Ensure nltk is imported here for download
nltk.download('averaged_perceptron_tagger_eng') # Download the required tagger, specifically 'eng'

def get_wordnet_pos(word):
    """Map NLTK POS tag to WordNet POS tag for lemmatization"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ, "N": wordnet.NOUN, "V": wordnet.VERB, "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

nltk_lemmatized_tokens = []
print("NLTK Lemmatization (WordNetLemmatizer with POS tags):")
for i, sentence_words in enumerate(nltk_word_tokens):
    lemmatized_sentence = [nltk_lemmatizer.lemmatize(word, get_wordnet_pos(word)) for word in sent
    nltk_lemmatized_tokens.append(lemmatized_sentence)
    print(f"Sentence {i+1} Lemmatized: {lemmatized_sentence}")
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
NLTK Lemmatization (WordNetLemmatizer with POS tags):
Sentence 1 Lemmatized: ['The', 'patient', 'present', 'with', 'severe', 'abdominal', 'pain', 'and',
Sentence 2 Lemmatized: ['A', 'preliminary', 'diagnosis', 'of', 'appendicitis', 'be', 'make', 'base
Sentence 3 Lemmatized: ['Further', 'investigation', ',', 'include', 'a', 'CT', 'scan', ',', 'be',
```

```
Sentence 4 Lemmatized: ['The', 'patient', 'have', 'a', 'history', 'of', 'gastroesophageal', 'reflu
```

Reasoning: I have completed all NLTK preprocessing steps (sentence tokenization, word tokenization, stemming, and lemmatization). The next part of the task is to perform tokenization and lemmatization using spaCy. I will start by loading the spaCy English model, which was previously downloaded, and then use it to process the medical text for tokenization.

```
nlp = spacy.load('en_core_web_sm')
spacy_doc = nlp(medical_text)

print("spaCy Tokenization (Sentence and Word combined):")
for i, sent in enumerate(spacy_doc.sents):
    words = [token.text for token in sent]
    print(f"Sentence {i+1} Words: {words}")
```

```
spaCy Tokenization (Sentence and Word combined):
Sentence 1 Words: ['The', 'patient', 'presented', 'with', 'severe', 'abdominal', 'pain', 'and', 'n
Sentence 2 Words: ['A', 'preliminary', 'diagnosis', 'of', 'appendicitis', 'was', 'made', 'based',
Sentence 3 Words: ['Further', 'investigation', ',', 'including', 'a', 'CT', 'scan', ',', 'is', 're
Sentence 4 Words: ['The', 'patient', 'has', 'a', 'history', 'of', 'gastroesophageal', 'reflux', 'd
```

Reasoning: Now that spaCy has tokenized the medical text, the next step is to perform lemmatization using spaCy. I will iterate through the `spacy_doc` object, which contains the processed tokens, and extract the lemma attribute for each token.

```
spacy_lemmatized_tokens = []
print("spaCy Lemmatization:")
for i, sent in enumerate(spacy_doc.sents):
    lemmas = [token.lemma_ for token in sent]
    spacy_lemmatized_tokens.append(lemmas)
    print(f"Sentence {i+1} Lemmatized: {lemmas}")
```

```
spaCy Lemmatization:
Sentence 1 Lemmatized: ['the', 'patient', 'present', 'with', 'severe', 'abdominal', 'pain', 'and',
Sentence 2 Lemmatized: ['a', 'preliminary', 'diagnosis', 'of', 'appendicitis', 'be', 'make', 'base
Sentence 3 Lemmatized: ['further', 'investigation', ',', 'include', 'a', 'CT', 'scan', ',', 'be',
Sentence 4 Lemmatized: ['the', 'patient', 'have', 'a', 'history', 'of', 'gastroesophageal', 'reflu
```

Reasoning: Now that both NLTK stemming and lemmatization have been performed, I will compare their outputs to highlight the differences between these two techniques for word normalization.

```
print("\nComparison of NLTK Stemming vs. NLTK Lemmatization:")
for i in range(len(nltk_word_tokens)):
    print(f"Sentence {i+1}:")
    for j in range(len(nltk_word_tokens[i])):
        original = nltk_word_tokens[i][j]
        stemmed = nltk_stemmed_tokens[i][j]
        lemmatized = nltk_lemmatized_tokens[i][j]
        if stemmed != lemmatized:
            print(f"  Original: {original:<15} | Stemmed: {stemmed:<15} | Lemmatized: {lemmatizec
```

```

Comparison of NLTK Stemming vs. NLTK Lemmatization:
Sentence 1:
  Original: The             | Stemmed: the             | Lemmatized: The
```

```
  Original: severe          | Stemmed: sever           | Lemmatized: severe
  Original: abdominal       | Stemmed: abdomin         | Lemmatized: abdominal
Sentence 2:
  Original: A               | Stemmed: a               | Lemmatized: A
  Original: preliminary     | Stemmed: preliminari     | Lemmatized: preliminary
  Original: diagnosis       | Stemmed: diagnosi        | Lemmatized: diagnosis
  Original: appendicitis    | Stemmed: append          | Lemmatized: appendicitis
  Original: was             | Stemmed: wa              | Lemmatized: be
  Original: made            | Stemmed: made            | Lemmatized: make
  Original: clinical        | Stemmed: clinic          | Lemmatized: clinical
  Original: elevated        | Stemmed: elev            | Lemmatized: elevate
Sentence 3:
  Original: Further         | Stemmed: further         | Lemmatized: Further
  Original: investigation   | Stemmed: investig        | Lemmatized: investigation
  Original: including       | Stemmed: includ          | Lemmatized: include
  Original: CT              | Stemmed: ct              | Lemmatized: CT
  Original: is              | Stemmed: is              | Lemmatized: be
  Original: diagnosis       | Stemmed: diagnosi        | Lemmatized: diagnosis
  Original: determine       | Stemmed: determin        | Lemmatized: determine
  Original: course          | Stemmed: cours           | Lemmatized: course
Sentence 4:
  Original: The             | Stemmed: the             | Lemmatized: The
  Original: has             | Stemmed: ha              | Lemmatized: have
  Original: history         | Stemmed: histori         | Lemmatized: history
  Original: gastroesophageal | Stemmed: gastroesophag   | Lemmatized: gastroesophageal
  Original: disease         | Stemmed: diseas          | Lemmatized: disease
  Original: GERD            | Stemmed: gerd            | Lemmatized: GERD
  Original: occasional      | Stemmed: occasion        | Lemmatized: occasional
  Original: migraines       | Stemmed: migrain         | Lemmatized: migraine
  Original: are             | Stemmed: are             | Lemmatized: be
  Original: currently       | Stemmed: current         | Lemmatized: currently
  Original: well-managed    | Stemmed: well-manag      | Lemmatized: well-managed
  Original: medication      | Stemmed: medic           | Lemmatized: medication
```

Reasoning: Now that both NLTK and spaCy lemmatization have been performed, I will compare their outputs to highlight the differences and similarities between these two libraries for lemmatization.

```
print("\nComparison of NLTK Lemmatization vs. spaCy Lemmatization:")
for i in range(len(nltk_word_tokens)):
    print(f"Sentence {i+1}:")
    for j in range(len(nltk_word_tokens[i])):
        original = nltk_word_tokens[i][j]
        nltk_lemma = nltk_lemmatized_tokens[i][j]
        # Ensure spaCy tokens exist for comparison, handling potential length differences
        # due to different tokenization (e.g., well-managed vs well- - managed)
        spacy_lemma = "N/A"
        if i < len(spacy_lemmatized_tokens) and j < len(spacy_lemmatized_tokens[i]):
            spacy_lemma = spacy_lemmatized_tokens[i][j]

        if nltk_lemma != spacy_lemma:
            print(f"  Original: {original:<15} | NLTK Lemma: {nltk_lemma:<15} | spaCy Lemma: {spac
```

```
Comparison of NLTK Lemmatization vs. spaCy Lemmatization:
Sentence 1:
  Original: The             | NLTK Lemma: The             | spaCy Lemma: the
Sentence 2:
  Original: A               | NLTK Lemma: A               | spaCy Lemma: a
Sentence 3:
  Original: Further         | NLTK Lemma: Further         | spaCy Lemma: further
Sentence 4:
  Original: The             | NLTK Lemma: The             | spaCy Lemma: the
  Original: well-managed    | NLTK Lemma: well-managed    | spaCy Lemma: well
```

```
Original: with          | NLTK Lemma: with          | spaCy Lemma: -
Original: medication    | NLTK Lemma: medication    | spaCy Lemma: manage
Original: .             | NLTK Lemma: .             | spaCy Lemma: with
```

Final Task

Subtask:

Summarize the observations from comparing stemming and lemmatization, discussing how each method handles word forms and their implications for text analysis, especially in a medical context.

Summary:

Data Analysis Key Findings

- **NLTK Stemming (Porter Stemmer)**: This method aggressively reduces words to their root form (stem), which may not always be a linguistically valid word. For example, "preliminary" was stemmed to "preliminari", "diagnosis" to "diagnosi", "appendicitis" to "append", and "was" to "wa". This approach focuses on morphological variants but can result in non-dictionary words.
- **NLTK Lemmatization (WordNetLemmatizer with POS tags)**: This method produces the base or dictionary form (lemma) of a word, ensuring the output is a valid word. It uses Part-of-Speech (POS) tagging to improve accuracy. For instance, "was" was lemmatized to "be", "made" to "make", and "migraines" to "migraine". This approach maintains semantic meaning better than stemming.
- **spaCy Lemmatization**: This method also produces linguistically valid lemmas and is generally robust. It handled tokenization of complex terms like "well-managed" by splitting it into "well", "-", "managed" and then lemmatizing each part. Differences from NLTK lemmatization were noted, particularly regarding capitalization (spaCy often lowercases) and tokenization granularity.
- **Comparison of Stemming vs. Lemmatization**: Stemming (e.g., Porter Stemmer) is typically more aggressive, often truncating words to non-dictionary forms and is faster. Lemmatization, on the other hand, is more linguistically sophisticated, reducing words to their base dictionary form and often requiring POS tagging, thus being more computationally intensive but producing more semantically meaningful results. For example, "severe" was stemmed to "sever" but lemmatized to "severe".
- **Comparison of NLTK Lemmatization vs. spaCy Lemmatization**: While both aim to find the base form, spaCy often exhibits more advanced tokenization (e.g., handling hyphenated words granularly) and tends to lowercase lemmas more consistently. NLTK's lemmatizer might preserve capitalization and tokenization as-is more often.

Insights or Next Steps

- For medical text analysis, lemmatization (especially with POS tagging from NLTK or spaCy) is generally preferred over stemming. This is because medical terminology requires high precision; reducing "diagnosis" to "diagnosi" or "migraines" to "migraine" through stemming could lose specific clinical meaning or make the terms harder to interpret for downstream tasks like information extraction or concept linking.
- When choosing between NLTK and spaCy for lemmatization in a medical context, spaCy's integrated tokenization and robust POS tagging might offer a more streamlined and potentially accurate solution

for handling complex medical language and compound terms. Further evaluation with a larger, domain-specific medical corpus would be beneficial to determine the best approach.