

⌵ B I <> ↻ 🖼️ 🔍 ⌵ ⌵ — 📄 😊 🗑️ Close

2403A52018
GUJJA RAJU
BATCH - 02 CSE AIML

2403A52018 GUJJA RAJU BATCH - 02 CSE AIML

```
!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) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3)
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-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.15)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.13)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (8.3.10)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.5.2)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.4.3)
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.20.0)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.32.4)
Requirement already satisfied: pydantic!=1.8,!<1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.11.10)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (25.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!<1.8.1,<3.0.0,>=1.7.4) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!<1.8.1,<3.0.0,>=1.7.4) (2.41.4)
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!<1.8.1,<3.0.0,>=1.7.4) (4.14.1)
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!<1.8.1,<3.0.0,>=1.7.4) (0.4.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0) (3.10.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0) (2025.11.11)
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4) (1.3.0)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4) (0.0.1)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2) (0.19.0)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2) (7.0.5)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from Jinja2) (3.0.3)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open<8.0.0,>=5.2.1) (1.17.0)
```

```
import nltk
import spacy
```

```
medical_text = """
Diabetes is a chronic disease that affects how the body processes blood sugar.
If untreated, diabetes may cause heart disease, kidney failure, nerve damage and vision problems.
Early diagnosis and proper treatment help improve patient outcomes.
"""
```

```
print(medical_text)
```

```
Diabetes is a chronic disease that affects how the body processes blood sugar.
If untreated, diabetes may cause heart disease, kidney failure, nerve damage and vision problems.
Early diagnosis and proper treatment help improve patient outcomes.
```

Start coding or [generate](#) with AI.

Task

Perform Natural Language Processing (NLP) analysis on the provided medical text using NLTK and spaCy, including downloading necessary data/models, tokenization, stemming, and lemmatization, then compare their outputs, and provide a discussion on the importance of lemmatization in healthcare NLP.

✓ Verify Medical Text

Subtask:

Confirm that the `medical_text` variable is correctly loaded and contains the provided medical corpus.

Reasoning: To confirm that the `medical_text` variable is correctly loaded and contains the provided medical corpus, I will display its content.

```
print(medical_text)
```

```
Diabetes is a chronic disease that affects how the body processes blood sugar.  
If untreated, diabetes may cause heart disease, kidney failure, nerve damage and vision problems.  
Early diagnosis and proper treatment help improve patient outcomes.
```

✓ Download NLTK Data and SpaCy Model

Subtask:

Download necessary NLTK data (punkt, wordnet) and the 'en_core_web_sm' spaCy model for tokenization and lemmatization.

Reasoning: To download the 'punkt' and 'wordnet' datasets required for NLTK, I will use the `nltk.download()` function.

```
nltk.download('punkt')  
nltk.download('wordnet')  
print("NLTK 'punkt' and 'wordnet' data downloaded.")
```

```
[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 'punkt' and 'wordnet' data downloaded.
```

Reasoning: The previous step successfully downloaded the NLTK data. Now, to complete the subtask, I need to download the 'en_core_web_sm' spaCy model using `spacy.cli.download()`.

```
import spacy  
import subprocess  
  
try:  
    spacy.load("en_core_web_sm")  
    print("spaCy model 'en_core_web_sm' is already downloaded.")  
except OSError:  
    print("Downloading spaCy model 'en_core_web_sm'...")  
    subprocess.run(["python", "-m", "spacy", "download", "en_core_web_sm"], check=True)  
    print("spaCy model 'en_core_web_sm' downloaded.")
```

```
spaCy model 'en_core_web_sm' is already downloaded.
```

✓ Tokenize with NLTK

Subtask:

Tokenize the `medical_text` into sentences and then into words using NLTK's `sent_tokenize` and `word_tokenize`.

Reasoning: To tokenize the `medical_text` into sentences and words using NLTK, I will import the necessary functions `sent_tokenize` and `word_tokenize` and then apply them to the text, storing the results in `nltk_sentences` and `nltk_words` respectively, before printing both lists.

```
from nltk.tokenize import sent_tokenize, word_tokenize  
  
# Tokenize into sentences  
nltk_sentences = sent_tokenize(medical_text)
```

```
# Tokenize into words
nltk_words = word_tokenize(medical_text)




print("NLTK Sentences:", nltk_sentences)
print("\nNLTK Words:", nltk_words)
```

```
NLTK Sentences: ['\nDiabetes is a chronic disease that affects how the body processes blood sugar.', 'If untreated, diabetes may\n']
NLTK Words: ['Diabetes', 'is', 'a', 'chronic', 'disease', 'that', 'affects', 'how', 'the', 'body', 'processes', 'blood', 'sugar']
```

```
import pandas as pd

comparison_df = pd.DataFrame({
    'NLTK Original Words': nltk_words,
    'NLTK Stemmed Words': nltk_stemmed_words,
    'NLTK Lemmatized Words': nltk_lemmatized_words,
    'spaCy Lemmatized Words': spacy_lemmatized_words
})

display(comparison_df)
```

	NLTK Original Words	NLTK Stemmed Words	NLTK Lemmatized Words	spaCy Lemmatized Words	
0	Diabetes	diabet	Diabetes	Diabetes	
1	is	is	is	be	
2	a	a	a	a	
3	chronic	chronic	chronic	chronic	
4	disease	diseas	disease	disease	
5	that	that	that	that	
6	affects	affect	affect	affect	
7	how	how	how	how	
8	the	the	the	the	
9	body	bodi	body	body	
10	processes	process	process	process	
11	blood	blood	blood	blood	
12	sugar	sugar	sugar	sugar	
13	
14	If	if	If	if	
15	untreated	untreat	untreated	untreat	
16	,	,	,	,	
17	diabetes	diabet	diabetes	diabete	
18	may	may	may	may	
19	cause	caus	cause	cause	
20	heart	heart	heart	heart	
21	disease	diseas	disease	disease	
22	,	,	,	,	
23	kidney	kidney	kidney	kidney	
24	failure	failur	failure	failure	
25	,	,	,	,	
26	nerve	nerv	nerve	nerve	
27	damage	damag	damage	damage	
28	and	and	and	and	
29	vision	vision	vision	vision	
30	problems	problem	problem	problem	
31	
32	Early	earli	Early	early	
33	diagnosis	diagnosi	diagnosis	diagnosis	
34	and	and	and	and	
35	proper	proper	proper	proper	
36	treatment	treatment	treatment	treatment	
37	help	help	help	help	
38	improve	improv	improve	improve	
39	patient	patient	patient	patient	
40	outcomes	outcom	outcome	outcome	
41	

Next steps:

[Generate code with comparison_df](#)[New interactive sheet](#)

The table above presents a side-by-side comparison of the original tokens, NLTK-stemmed words, NLTK-lemmatized words, and spaCy-lemmatized words. This format allows for easy observation of the differences and similarities between stemming and lemmatization, as well as between NLTK's and spaCy's lemmatization approaches.

Reasoning: The previous cell failed because the 'punkt_tab' resource required by NLTK's sentence tokenizer was not found. To fix this, I need to explicitly download the 'punkt_tab' resource using `nltk.download('punkt_tab')` before performing tokenization, as suggested by the error message.

```
from nltk.tokenize import sent_tokenize, word_tokenize
import nltk # Ensure nltk is imported for download function

# Download the missing 'punkt_tab' resource
nltk.download('punkt_tab')

# Tokenize into sentences
nltk_sentences = sent_tokenize(medical_text)

# Tokenize into words
nltk_words = word_tokenize(medical_text)

print("NLTK Sentences:", nltk_sentences)
print("\nNLTK Words:", nltk_words)
```

[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt_tab.zip.
NLTK Sentences: ['\nDiabetes is a chronic disease that affects how the body processes blood sugar.', 'If untreated, diabetes may
NLTK Words: ['Diabetes', 'is', 'a', 'chronic', 'disease', 'that', 'affects', 'how', 'the', 'body', 'processes', 'blood', 'sugar']

Reasoning: Now that the NLTK tokenization is complete, the next step is to tokenize the `medical_text` using spaCy. I will load the 'en_core_web_sm' model, process the text to create a Doc object, and then extract and print the tokens from it.

```
import spacy

# Load the spaCy English model
nlp = spacy.load("en_core_web_sm")

# Process the medical text with spaCy
spacy_doc = nlp(medical_text)

# Extract tokens
spacy_tokens = [token.text for token in spacy_doc]

print("spaCy Tokens:", spacy_tokens)
```

spaCy Tokens: ['\n', 'Diabetes', 'is', 'a', 'chronic', 'disease', 'that', 'affects', 'how', 'the', 'body', 'processes', 'blood',

✓ Stemming with NLTK

Subtask:

Perform stemming on the `nltk_words` list using NLTK's Porter Stemmer.

Reasoning: To perform stemming using NLTK's Porter Stemmer, I first need to import the `PorterStemmer` and then iterate through the `nltk_words` to apply stemming to each word, storing the results in `nltk_stemmed_words` and printing them.

```
from nltk.stem import PorterStemmer

# Initialize Porter Stemmer
porter_stemmer = PorterStemmer()

# Perform stemming
nltk_stemmed_words = [porter_stemmer.stem(word) for word in nltk_words]

print("NLTK Stemmed Words:", nltk_stemmed_words)
```

NLTK Stemmed Words: ['diabet', 'is', 'a', 'chronic', 'diseas', 'that', 'affect', 'how', 'the', 'bodi', 'process', 'blood', 'suga

✓ Lemmatization with NLTK

Subtask:

Perform lemmatization on the `nltk_words` list using NLTK's WordNetLemmatizer.

Reasoning: To perform lemmatization using NLTK's WordNetLemmatizer, I need to import the `WordNetLemmatizer` class, initialize it, and then iterate through the `nltk_words` list to apply lemmatization to each word, storing the results in `nltk_lemmatized_words` and printing them.

```
from nltk.stem import WordNetLemmatizer

# Initialize WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

# Perform lemmatization
nltk_lemmatized_words = [wordnet_lemmatizer.lemmatize(word) for word in nltk_words]

print("NLTK Lemmatized Words:", nltk_lemmatized_words)
```

NLTK Lemmatized Words: ['Diabetes', 'is', 'a', 'chronic', 'disease', 'that', 'affect', 'how', 'the', 'body', 'process', 'blood',

Lemmatization with spaCy

Subtask:

Perform lemmatization on the `spacy_doc` object using spaCy's built-in lemmatizer.

✓ Lemmatization with spaCy

Subtask:

Perform lemmatization on the `spacy_doc` object using spaCy's built-in lemmatizer.

Reasoning: To perform lemmatization with spaCy, I will iterate through the `spacy_doc` object, accessing the `lemma_` attribute for each token, and store the results in a list named `spacy_lemmatized_words` before printing them.

```
import spacy

# Ensure the nlp object is loaded
# This line is redundant if nlp is already in scope from previous cells, but good for self-contained execution
if 'nlp' not in locals():
    nlp = spacy.load("en_core_web_sm")

# If spacy_doc is not in scope, re-process the text
if 'spacy_doc' not in locals():
    spacy_doc = nlp(medical_text)

# Perform lemmatization using spaCy
spacy_lemmatized_words = [token.lemma_ for token in spacy_doc if not token.is_space]

print("spaCy Lemmatized Words:", spacy_lemmatized_words)
```

spaCy Lemmatized Words: ['Diabetes', 'be', 'a', 'chronic', 'disease', 'that', 'affect', 'how', 'the', 'body', 'process', 'blood']

✓ Comparison of NLTK and spaCy Lemmatization and Importance in Healthcare NLP

Subtask:

Compare the lemmatized outputs from NLTK and spaCy, then provide a discussion on the importance of lemmatization in healthcare NLP.

Reasoning: To facilitate the comparison of NLTK and spaCy lemmatized outputs, I will first print both lists for easy visual inspection, which will help in identifying their differences.

```
print("NLTK Lemmatized Words:", nltk_lemmatized_words)
print("\nspacey Lemmatized Words:", spacy_lemmatized_words)
```

```
NLTK Lemmatized Words: ['Diabetes', 'is', 'a', 'chronic', 'disease', 'that', 'affect', 'how', 'the', 'body', 'process', 'blood',
spacey Lemmatized Words: ['Diabetes', 'be', 'a', 'chronic', 'disease', 'that', 'affect', 'how', 'the', 'body', 'process', 'blood']
```

Discussion on the Importance of Lemmatization in Healthcare NLP

Comparing the outputs, we can observe subtle but significant differences between NLTK's `WordNetLemmatizer` and spaCy's built-in lemmatizer:

- **"is" vs. "be":** NLTK lemmatizes "is" to "is", while spaCy correctly lemmatizes it to "be". This is a common difference, as spaCy often has more sophisticated linguistic models that can handle verb conjugations more accurately.
- **Case Sensitivity:** NLTK's `WordNetLemmatizer` is generally case-sensitive by default (e.g., 'If' remains 'If'). spaCy, on the other hand, often normalizes to lowercase unless explicitly configured otherwise (e.g., 'If' becomes 'if', 'Early' becomes 'early'). For words like 'Diabetes', both retain the capitalization.
- **Consistency:** spaCy generally provides more consistent and context-aware lemmatization due to its statistical models and part-of-speech (POS) tagging capabilities. For instance, "untreated" is lemmatized to "untreated" by NLTK, but to "untreat" by spaCy, and "diabetes" remains "diabetes" in NLTK while spaCy gives "diabete". This can be critical for downstream tasks.

Importance in Healthcare NLP:

Lemmatization is crucial in healthcare NLP for several reasons:

1. **Standardization of Medical Terminology:** Medical records and research papers often contain variations of the same medical term (e.g., "diagnosed", "diagnosing", "diagnosis"). Lemmatization reduces these terms to their base form ("diagnosis"), allowing for more accurate counting, indexing, and retrieval of information, regardless of grammatical variations.
2. **Improved Information Retrieval:** When searching for information about a specific condition or treatment, lemmatization ensures that all relevant documents are found, even if they use different inflections of the search terms. For example, a search for "treat" would also find documents containing "treatment," "treated," or "treatments."
3. **Enhanced Data Analysis:** In tasks like sentiment analysis of patient feedback, lemmatization helps aggregate similar opinions. "Recovered" and "recovering" would both be reduced to "recover", providing a clearer picture of patient outcomes or experiences.
4. **Feature Engineering for Machine Learning:** For machine learning models trained on medical text (e.g., predicting disease outbreaks, identifying adverse drug reactions), lemmatization reduces the sparsity of the vocabulary. By mapping inflected words to their base forms, the models can generalize better and learn more robust patterns, as they are not treating different forms of the same word as distinct features.
5. **Clinical Decision Support Systems:** Accurate lemmatization can improve the performance of systems that assist clinicians by providing relevant information from vast medical literature, ensuring that all related concepts are linked.
6. **Reduced Ambiguity:** While stemming can sometimes produce non-dictionary words (e.g., "diseas"), lemmatization aims to return a valid word, which is particularly important in a domain where precision and clarity are paramount.

In conclusion, while both NLTK and spaCy offer lemmatization, spaCy's context-aware approach often provides more linguistically accurate lemmas, which is highly beneficial for the complex and sensitive nature of medical text analysis. Accurate lemmatization directly contributes to better understanding, analysis, and application of information in the healthcare domain.

Discussion on the Importance of Lemmatization in Healthcare NLP

Comparing the outputs, we can observe subtle but significant differences between NLTK's `WordNetLemmatizer` and spaCy's built-in lemmatizer:

- **"is" vs. "be":** NLTK lemmatizes "is" to "is", while spaCy correctly lemmatizes it to "be". This is a common difference, as spaCy often has more sophisticated linguistic models that can handle verb conjugations more accurately.
- **Case Sensitivity:** NLTK's `WordNetLemmatizer` is generally case-sensitive by default (e.g., 'If' remains 'If'). spaCy, on the other hand, often normalizes to lowercase unless explicitly configured otherwise (e.g., 'If' becomes 'if', 'Early' becomes 'early'). For words like 'Diabetes', both retain the capitalization.
- **Consistency:** spaCy generally provides more consistent and context-aware lemmatization due to its statistical models and part-of-speech (POS) tagging capabilities. For instance, "untreated" is lemmatized to "untreated" by NLTK, but to "untreat" by spaCy, and

"diabetes" remains "diabetes" in NLTK while spaCy gives "diabete". This can be critical for downstream tasks.

Importance in Healthcare NLP:

Lemmatization is crucial in healthcare NLP for several reasons:

1. **Standardization of Medical Terminology:** Medical records and research papers often contain variations of the same medical term (e.g., "diagnosed", "diagnosing", "diagnosis"). Lemmatization reduces these terms to their base form ("diagnosis"), allowing for more accurate counting, indexing, and retrieval of information, regardless of grammatical variations.
2. **Improved Information Retrieval:** When searching for information about a specific condition or treatment, lemmatization ensures that all relevant documents are found, even if they use different inflections of the search terms. For example, a search for "treat" would also find documents containing "treatment," "treated," or "treatments."
3. **Enhanced Data Analysis:** In tasks like sentiment analysis of patient feedback, lemmatization helps aggregate similar opinions. "Recovered" and "recovering" would both be reduced to "recover", providing a clearer picture of patient outcomes or experiences.
4. **Feature Engineering for Machine Learning:** For machine learning models trained on medical text (e.g., predicting disease outbreaks, identifying adverse drug reactions), lemmatization reduces the sparsity of the vocabulary. By mapping inflected words to their base forms, the models can generalize better and learn more robust patterns, as they are not treating different forms of the same word as distinct features.
5. **Clinical Decision Support Systems:** Accurate lemmatization can improve the performance of systems that assist clinicians by providing relevant information from vast medical literature, ensuring that all related concepts are linked.
6. **Reduced Ambiguity:** While stemming can sometimes produce non-dictionary words (e.g., "diseas"), lemmatization aims to return a valid word, which is particularly important in a domain where precision and clarity are paramount.

In conclusion, while both NLTK and spaCy offer lemmatization, spaCy's context-aware approach often provides more linguistically accurate lemmas, which is highly beneficial for the complex and sensitive nature of medical text analysis. Accurate lemmatization directly contributes to better understanding, analysis, and application of information in the healthcare domain.

▼ Compare Outputs

Subtask:

Display and compare the outputs of tokenization, stemming, and lemmatization from both NLTK and spaCy for a clear understanding.

Reasoning: To display and compare all the outputs, I will first print the NLTK tokenized sentences as requested in the instructions.

```
print("NLTK Tokenized Sentences:", nltk_sentences)
```

```
NLTK Tokenized Sentences: ['\nDiabetes is a chronic disease that affects how the body processes blood sugar.', 'If untreated, di
```

Reasoning: To continue displaying the outputs for comparison, I will now print the NLTK tokenized words, as per the instructions.

```
print("NLTK Tokenized Words:", nltk_words)
```

```
NLTK Tokenized Words: ['Diabetes', 'is', 'a', 'chronic', 'disease', 'that', 'affects', 'how', 'the', 'body', 'processes', 'blood
```

Reasoning: To continue displaying the outputs for comparison, I will now print the spaCy tokenized words, as per the instructions.

```
print("spaCy Tokenized Words:", spacy_tokens)
```

```
spaCy Tokenized Words: ['\n', 'Diabetes', 'is', 'a', 'chronic', 'disease', 'that', 'affects', 'how', 'the', 'body', 'processes',
```

Reasoning: To continue displaying the outputs for comparison, I will now print the NLTK stemmed words, as per the instructions.

```
print("NLTK Stemmed Words:", nltk_stemmed_words)
```

```
NLTK Stemmed Words: ['diabet', 'is', 'a', 'chronic', 'diseas', 'that', 'affect', 'how', 'the', 'bodi', 'process', 'blood', 'suga
```

Reasoning: To continue displaying the outputs for comparison, I will now print the NLTK lemmatized words, as per the instructions.

```
print("NLTK Lemmatized Words:", nltk_lemmatized_words)
```



```
NLTK Lemmatized Words: ['Diabetes', 'is', 'a', 'chronic', 'disease', 'that', 'affect', 'how', 'the', 'body', 'process', 'blood',
```

Reasoning: To continue displaying the outputs for comparison, I will now print the spaCy lemmatized words, as per the instructions.

```
print("spaCy Lemmatized Words:", spacy_lemmatized_words)
```

```
spaCy Lemmatized Words: ['Diabetes', 'be', 'a', 'chronic', 'disease', 'that', 'affect', 'how', 'the', 'body', 'process', 'blood']
```

Comparative Analysis of NLTK vs. spaCy for Tokenization, Stemming, and Lemmatization

Having printed the outputs for tokenization, stemming, and lemmatization using both NLTK and spaCy, we can now conduct a comparative analysis:

1. Tokenization

- **NLTK (nltk_sentences, nltk_words):**

- `sent_tokenize`: Successfully segments the text into sentences, handling punctuation boundaries. It retains the leading newline character for the first sentence, which might require pre-processing if not desired.
- `word_tokenize`: Breaks sentences into words and punctuation marks. It handles contractions and complex words reasonably well, treating punctuation as separate tokens.

- **spaCy (spacy_tokens):**

- spaCy's tokenizer processes the entire document, and its tokens often include whitespace characters as separate tokens, as seen with `['\n']` at the beginning and end of lines. This is a design choice in spaCy, treating all components as tokens.
- spaCy generally produces more refined tokens, often splitting contractions (e.g., "don't" -> "do", "n't") or handling more complex punctuation scenarios based on its statistical models. In this specific `medical_text`, both NLTK and spaCy produced similar word tokens, except for spaCy including the newline characters and NLTK treating them as part of the first sentence's leading whitespace.

2. Stemming (NLTK's Porter Stemmer - nltk_stemmed_words)

- **NLTK Stemming:** The Porter Stemmer reduces words to their root form, which is often an artificially shortened version that might not be a valid word in itself (e.g., "disease" -> "diseas", "affects" -> "affect", "body" -> "bodi", "early" -> "earli", "problems" -> "problem"). While effective for information retrieval by grouping related words, its output can be less readable and sometimes ambiguous due to the lack of linguistic context.

- Example: `['diabet', 'is', 'a', 'chronic', 'diseas', 'that', 'affect', ...]`

3. Lemmatization

- **NLTK (WordNetLemmatizer - nltk_lemmatized_words):**

- NLTK's `WordNetLemmatizer` aims to return the base form (lemma) of a word, which is a valid dictionary word. However, its effectiveness often depends on providing the correct Part-of-Speech (POS) tag, which was not explicitly done here. Without POS tags, it defaults to noun lemmatization, which can lead to less accurate results for verbs or adjectives.
- Observations: "is" remains "is", "affects" becomes "affect", "processes" becomes "process", "problems" becomes "problem", "outcomes" becomes "outcome". "untreated" remains "untreated".

- **spaCy (spacy_lemmatized_words):**

- spaCy's lemmatizer is highly sophisticated, leveraging its statistical models and built-in POS tagger to provide context-aware and linguistically accurate lemmas. It tends to produce more consistent and often more correct base forms.
- Observations: "is" becomes "be" (correct lemma for the verb 'to be'), "affects" becomes "affect", "processes" becomes "process", "problems" becomes "problem", "untreated" becomes "untreat", "diabetes" becomes "diabete", "early" becomes "early", "outcomes" becomes "outcome".

Key Differences and Similarities:

- **Tokenization:** Both are effective, but spaCy includes whitespace tokens, and NLTK's `sent_tokenize` tends to include leading whitespace of a sentence with the sentence itself. For word tokens, they are largely similar for simple text.
- **Stemming vs. Lemmatization:** Stemming (NLTK) is a crude heuristic process that chops off suffixes, often resulting in non-words. Lemmatization (NLTK and spaCy) aims for actual dictionary words. Lemmatization is generally preferred for applications requiring higher linguistic accuracy.

- **Lemmatization Accuracy:** spaCy's lemmatizer is generally more robust and accurate due to its deep linguistic models and automatic POS tagging. This is evident in examples like "is" -> "be" and more consistent normalization (e.g., to lowercase for some tokens), while NLTK's `WordNetLemmatizer` often requires explicit POS tagging for optimal performance.
- **Ease of Use:** spaCy often provides a more integrated and streamlined NLP pipeline, where tokenization, POS tagging, and lemmatization are part of a single `nlp(text)` process, making it simpler to use for advanced linguistic tasks compared to NLTK's more modular, function-based approach.

In summary, while NLTK offers a good set of tools for basic NLP tasks, spaCy provides a more advanced, context-aware, and often more accurate solution, particularly for lemmatization, which is crucial for applications where linguistic precision is paramount.