In [ ]:
```
!pip install PdfReader
```

In [ ]:
```
!pip install pypdf
```

In [ ]:
```
!pip install nltk
```

In [ ]:
```
pip install --no-cache-dir spacy
```

In [ ]:
```
import pypdf
from pypdf import PdfReader
```

In [ ]:
```
reader = PdfReader('C:/Users/tarmc/Downloads/Gwen_Walz_Article.pdf')
reader
```

In [ ]:
```
# Part 1
from collections import Counter
import re

# Open the PDF file
with open('C:/Users/tarmc/Downloads/Gwen_Walz_Article.pdf', 'rb') as file:
    reader = pypdf.PdfReader(file)
    text = ''
    for page_number in range(len(reader.pages)):
        text += reader.pages[page_number].extract_text()

# Tokenize the text and count the occurrences of each word
words = re.findall(r'\b\w+\b', text.lower())
word_counts = Counter(words)

# Show the most common words
most_common_words = word_counts.most_common(10)  # Change 10 to the desired nu
print(most_common_words)
```

In [ ]:
```
# Part 2
import nltk
# Download the required nltk data (if not already downloaded)
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

# Open the PDF file
with open('C:/Users/tarmc/Downloads/Gwen_Walz_Article.pdf', 'rb') as file:
    reader = pypdf.PdfReader(file)
    text = ''
    for page_number in range(len(reader.pages)):
        text += reader.pages[page_number].extract_text()

# Tokenize the text and perform part-of-speech tagging
sentences = nltk.sent_tokenize(text)
tagged_words = [nltk.pos_tag(nltk.word_tokenize(sentence)) for sentence in sen

# Filter words based on part of speech and count the occurrences of each word
noun_counts = Counter(word.lower() for sentence in tagged_words for word, tag

# Show the most common nouns
most_common_nouns = noun_counts.most_common(10)  # Change 10 to the desired nu
print(most_common_nouns)
```

In [ ]:
```
# Part 3
import spacy

# Load the English tokenizer, tagger, parser, NER, and word vectors
nlp = spacy.load("en_core_web_sm")

# Example sentence
sentence = "They married in 1994 and share two children Hope and Gus."

# Process the sentence using spaCy
doc = nlp(sentence)

# Extract subject/object relationships
subject_object_pairs = []
for token in doc:
    if token.dep_ in ["nsubj", "dobj"]:  # nsubj: nominal subject, dobj: direc
        subject_object_pairs.append((token.head.text, token.text, token.dep_))

print(subject_object_pairs)
```

In [ ]:
```
# Part 4
# Load the English tokenizer, tagger, parser, NER, and word vectors
nlp = spacy.load("en_core_web_sm")

# Open the PDF file
with open('C:/Users/tarmc/Downloads/Gwen_Walz_Article.pdf', 'rb') as file:
    reader = PyPDF2.PdfReader(file)
    text = ''
    for page_number in range(len(reader.pages)):
        text += reader.pages[page_number].extract_text()

# Process the text using spaCy
doc = nlp(text)

# Extract entities and their types
entity_types = Counter((ent.text, ent.label_) for ent in doc.ents)

# Show the most common entities and their types
most_common_entities = entity_types.most_common(10)  # Change 10 to the desire
print(most_common_entities)
```

In [ ]:
```
# Part 5
# Load the English tokenizer, tagger, parser, NER, and word vectors
nlp = spacy.load("en_core_web_sm")

# Open the PDF file
with open('Gwen_Walz_Article.pdf', 'rb') as file:
    reader = PyPDF2.PdfReader(file)
    text = ''
    for page_number in range(len(reader.pages)):
        text += reader.pages[page_number].extract_text()

# Process the text using spaCy
doc = nlp(text)

# Extract entities and their dependencies
entity_dependencies = [(ent.text, ent.label_, ent.root.head.text, ent.root.hea

# Show the entities and their dependencies
print(entity_dependencies)
```

In [ ]:
```
#Part 6
# Load the English tokenizer, tagger, parser, NER, and word vectors
nlp = spacy.load("en_core_web_md")  # Using the medium-sized model with word v

# Open the PDF file
with open('Gwen_Walz_Article.pdf', 'rb') as file:
    reader = PyPDF2.PdfReader(file)
    text = ''
    for page_number in range(len(reader.pages)):
        text += reader.pages[page_number].extract_text()

# Process the text using spaCy
doc = nlp(text)

# Find the most similar words
target_word = "climate"  # Change to the target word for which you want to find
similar_words = []
for token in doc:
    if token.is_alpha and token.text.lower() != target_word:  # Exclude the ta
        similarity = nlp(target_word).similarity(token)
        similar_words.append((token.text, similarity))

# Sort the similar words by similarity score
similar_words.sort(key=lambda x: x[1], reverse=True)

# Show the most similar words
print(similar_words[:10]) # Change 10 to the desired number of most similar wo
```