```
# Initialization for downloading NLTK packages (run once)
import nltk

# Download necessary NLTK packages (if not already installed)
nltk.download('punkt')
nltk.download('all')
print("NLTK packages downloaded successfully.")
```

Show hidden output

```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import sent_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import ipywidgets as widgets
from IPython.display import display, clear_output

# Function to extract text from a URL
def extract_text_from_url(url):
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'html.parser')
        paragraphs = soup.find_all('p')
        text = ' '.join([p.get_text() for p in paragraphs])
        return text
    except Exception as e:
        return f"Error extracting content: {e}"

# Function to find the best answer
def find_best_answer(question, text):
    sentences = sent_tokenize(text)
    if not sentences:
        return "No content available to extract an answer from."

    all_text = [question] + sentences
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(all_text)
    question_vector = tfidf_matrix[0]
    sentence_vectors = tfidf_matrix[1:]
    similarities = cosine_similarity(question_vector, sentence_vectors)
    best_sentence_index = similarities.argmax()
    best_answer = sentences[best_sentence_index]
    return best_answer

# Conversation storage
conversation_log = []

# Initial URL input
url_input = widgets.Text(
    value='',
    placeholder='Enter the initial URL here...',
    layout=widgets.Layout(width='100%')
)

# Conversation and question widgets
conversation_box = widgets.Textarea(
    value='',
    placeholder='Conversation history...',
    layout=widgets.Layout(height='300px', width='100%'),
    disabled=True
)

question_box = widgets.Text(
    value='',
    placeholder='Enter your question here and press Enter...',
    layout=widgets.Layout(width='100%')
)

# Display widgets
display(url_input)
display(conversation_box)
display(question_box)

# Function to handle URL submission
def on_url_submit(change):
    global text_content
    url = url_input.value.strip()
    if url:
        text_content = extract_text_from_url(url)
        print("\nContent extracted from the new URL.\n")
    else:
        print("\nInvalid URL. Please enter a valid URL.\n")

# Function to handle question submission
def on_question_submit(change):
    global text_content

    question = question_box.value.strip()  # Read and strip the input
    question_box.value = ''  # Clear the input box after submission

    if not question:
        return  # Ignore empty submissions

    if question.lower() == "change url":
        print("\nEnter the new URL in the URL input box above and press Enter.\n")
        return

    elif question.lower() == "exit":
        print("\nExiting conversation.")
        question_box.disabled = True
        return

    elif question.lower() == "clear":
        conversation_log.clear()
        conversation_box.value = ''
        print("\nConversation history cleared.\n")
        return

    answer = find_best_answer(question, text_content)
    conversation_log.append((question, answer))

    conversation_history = "\n".join([f"Q: {q}\nA: {a}" for q, a in conversation_log])
    conversation_box.value = conversation_history

# Attach event handlers
url_input.on_submit(on_url_submit)
question_box.on_submit(on_question_submit)
```

https://en.wikipedia.org/wiki/Deep_learning

Conversation history...

Enter your question here and press Enter...

```
Content extracted from the new URL.


Conversation history cleared.
```