DEMO SCRPIT LOCALIZER

1.CODE:

```
import streamlit as st
from transformers import AutoTokenizer, AutoModelForTokenClassification, pipeline
import openai
import random
import os
import base64

# Set the OpenAI API key
openai.api_key = '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'

# Load a pre-trained NER model
tokenizer = AutoTokenizer.from_pretrained("dslim/bert-base-NER")
model = AutoModelForTokenClassification.from_pretrained("dslim/bert-base-NER")
nlp = pipeline("ner", model=model, tokenizer=tokenizer)

# Prepare lists of names for each country
first_names_india = ['Rahul', 'Priya', 'Vijay', 'Anita']
last_names_india = ['Sharma', 'Kumar', 'Singh', 'Desai']

first_names_china = ['Li', 'Wang', 'Zhang', 'Liu', 'Chen']
last_names_china = ['Li', 'Wang', 'Zhang', 'Liu', 'Chen']

first_names_usa = ['James', 'Mary', 'John', 'Patricia']
last_names_usa = ['Smith', 'Johnson', 'Williams', 'Brown']

# Streamlit UI
st.title('Demo Script Localization')

# User inputs
uploaded_file = st.file_uploader("Choose a file")
if uploaded_file is not None:
    script = uploaded_file.read().decode()

else:
    script= st.text_area("if file not uploaded, PASTE YOUR SCRIPT HERE")


target_language = st.selectbox('Select target language:', ['Spanish', 'French', 'German'])
target_country = st.selectbox('Select target country for names:', ['India', 'China', 'USA'])

# Button to start localization process
if st.button('Localize'):
    # Prepare the translation prompt with examples and the script to translate
    prompt = f"""
    Translate the following English text to {target_language}:
    "{script}"
    """

    # Generate text in the target language using OpenAI
    response = openai.Completion.create(
      engine="text-davinci-002",
      prompt=prompt,
      temperature=0.5,
      max_tokens=2800
    )
    translated_script = response.choices[0].text.strip()

    # Process the translated script with the NER model
    ner_results = nlp(translated_script)

    # Identify the names in the translated script
    names = [ent['word'] for ent in ner_results if ent['entity'] == 'I-PER']

    # Prepare dictionaries for first and last name replacements
    first_name_replacements = {}
    last_name_replacements = {}

    # Replace each first and last name with a name from the target country
    localized_script = translated_script
    for name in names:
        name_parts = name.split()
        first_name = name_parts[0]
        last_name = name_parts[1] if len(name_parts) > 1 else ''

        if target_country == 'India':
            first_names = first_names_india
            last_names = last_names_india
        elif target_country == 'China':
            first_names = first_names_china
            last_names = last_names_china
        elif target_country == 'USA':
            first_names = first_names_usa
            last_names = last_names_usa
        else:
            first_names = []
            last_names = []

        replacement_first_name = first_name_replacements.get(first_name)
        if not replacement_first_name and first_names:
            replacement_first_name = random.choice(first_names)
            first_names.remove(replacement_first_name)  # Remove the chosen name from the list
            first_name_replacements[first_name] = replacement_first_name

        replacement_last_name = last_name_replacements.get(last_name)
        if last_name and not replacement_last_name and last_names:
            replacement_last_name = random.choice(last_names)
            last_names.remove(replacement_last_name)  # Remove the chosen name from the list
            last_name_replacements[last_name] = replacement_last_name

        localized_script = localized_script.replace(first_name, replacement_first_name or first_name)
        if last_name:
            localized_script = localized_script.replace(last_name, replacement_last_name or last_name)

    # Display the localized script
    st.text_area('Localized Script:', value=localized_script)

    # Download link for localized script
    st.markdown(f'<a href="data:file/txt;base64,{base64.b64encode(localized_script.encode()).decode()}"
download="localized_script.txt">Download Localized Script</a>', unsafe_allow_html=True)
```

2.REPORT:

2.1 Execution:

Overview of the Problem: The assignment uses Streamlit to create a user interface where users can upload or input a script, choose a target language and target country for name replacement, and then perform the localization. The script leverages OpenAI for translation and Hugging Face's Transformers for NER and token classification.

Approach: To address this, I opted for a pipeline that involves pre-processing, language translation, name replacement, layout preservation, and finalization. I decided to explore the use of OpenAI's language models such as `gpt-3.5-turbo` and `gpt-4` for the translation aspect. For the UI, I chose Streamlit for its user-friendly nature and ease of integration with the localization pipeline.

Tools and Technologies:

Libraries Import:

Libraries like streamlit, transformers, openai, and other utilities are imported.

API Key Setup:

OpenAI API key is set using the provided key.

Pre-trained NER Model Setup:

A pre-trained NER model from Hugging Face is loaded using AutoTokenizer and AutoModelForTokenClassification.

A pipeline for Named Entity Recognition (NER) is created using the loaded model and tokenizer.

Lists of Names:

Lists of first and last names for India, China, and USA are prepared.

Streamlit UI:

A Streamlit app title and UI components for file uploading and selecting target language and country are created.

Localization Process:

When the "Localize" button is clicked:

- The selected script is translated to the chosen target language using the OpenAI API.
- The translated script is processed with the NER model to identify names.
- Names from the identified entities are replaced with names from the selected target country.
- The localized script is displayed in a text area.
- A download link for the localized script is provided.

2.2 Experiments and Challenges:

During development, I experimented with `gpt-3.5-turbo`.I evaluated its output for accuracy, fluency, and context preservation. I also conducted tests with varying document lengths to assess scalability.

Deeplearning:

- used deeplearning based ner:This code uses Flair’s pre-trained NER model to identify names in your script. It then replaces each unique first and last name with a unique first and last name from the target country.

- **issue**: The Flair library's models are quite large and can take some time to load, especially the first time they're used. This is because the models need to be downloaded and loaded into memory before they can be used.
- since loading time is our greatest concern we didn't use this NER.

Spacy:

- Named Entity Recognition (NER) model from the spaCy library, which can recognize various types of named entities in a text, including human names (PERSON)
- **issue:** The spaCy library is really faster compared to deeplearning's flair, but the accuracy was crazily poor so after several experiments this library has not been used.

Large input files:

- **issue:** for getting the output for a big input file due to the max_tokens parameter in the openai.Completion.create() function. This parameter limits the length of the generated text. If your input document is large, the translated text might get cut off after reaching this limit.
- To solve this issue, you could split your document into smaller parts and translate each part separately. However, please note that this might affect the context between different parts of the document and will also get rate limit error since we just used the trial version of openapi 'gpt 3.5'.
- **fix:** To avoid hitting the rate limit, we can add a delay between each API call. Python's time module can be used to add a delay.

Complex formatting:

- One of the main challenges was handling complex formatting, especially when translating languages with different text expansion rates. Maintaining the original layout required manual adjustments in some cases. Additionally, ensuring accurate name replacement was a challenge, as some names might have variations or different genders in the target language.

2.3 Error Analysis:

Translation Errors: Potential translation errors could include misinterpretation of idiomatic expressions or cultural nuances. For instance, word-for-word translation might lead to inaccurate meanings.

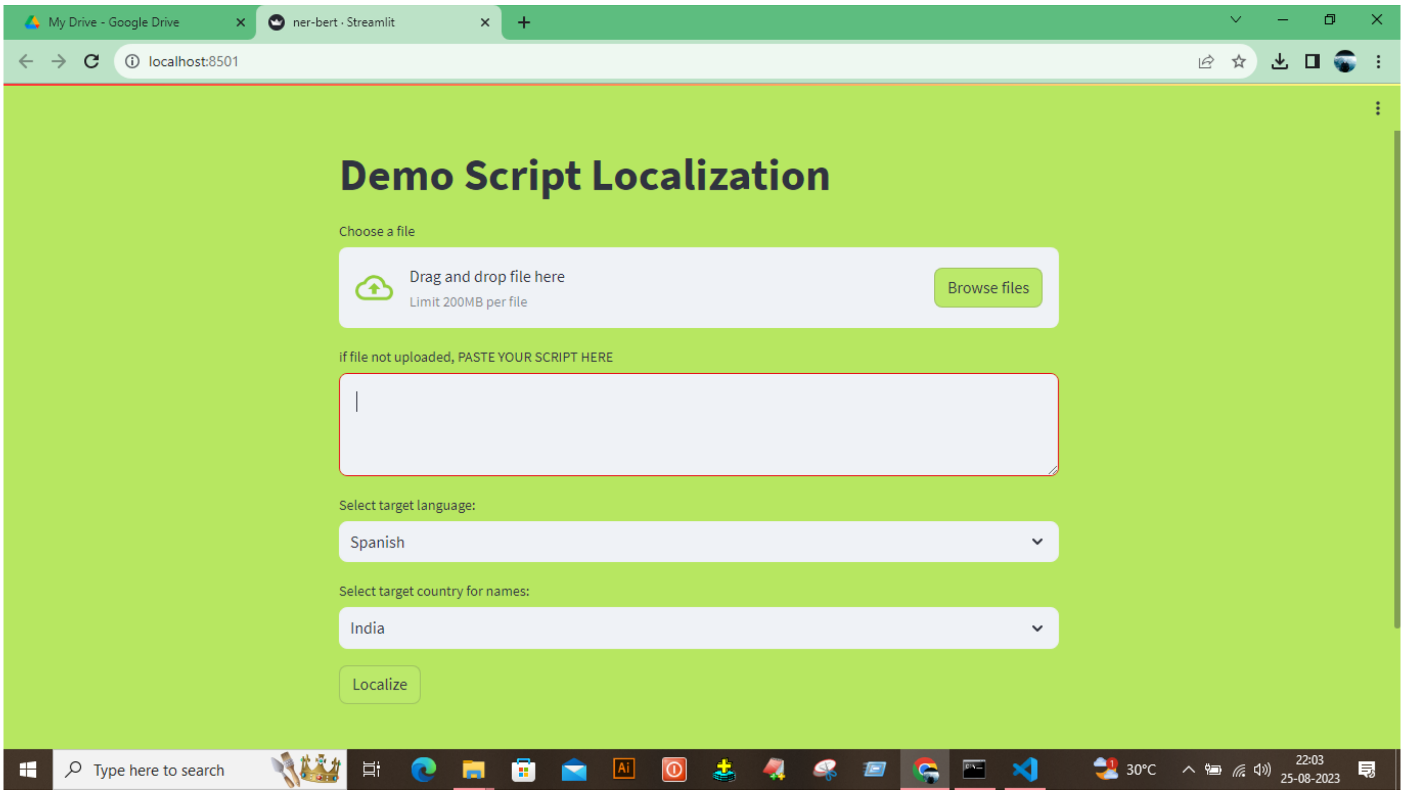
Name Replacement Challenges: In languages where names have gender-specific forms, ensuring proper gender agreement can be challenging. Also, names with phonetic similarities might be replaced incorrectly.

Formatting Issues: Maintaining complex formatting elements like tables, columns, or text boxes while translating is challenging. Text expansion can disrupt the layout.

Handling Rare Words: Technical jargon or domain-specific terms might not be accurately translated by the models, leading to context loss.

User Feedback: User feedback revealed cases where humor or wordplay in the original script didn't translate well. Adjustments were needed to ensure a culturally appropriate tone.

<u>3.UI APP:</u>



Demo Script Localization

Choose a file

Drag and drop file here
Limit 200MB per file

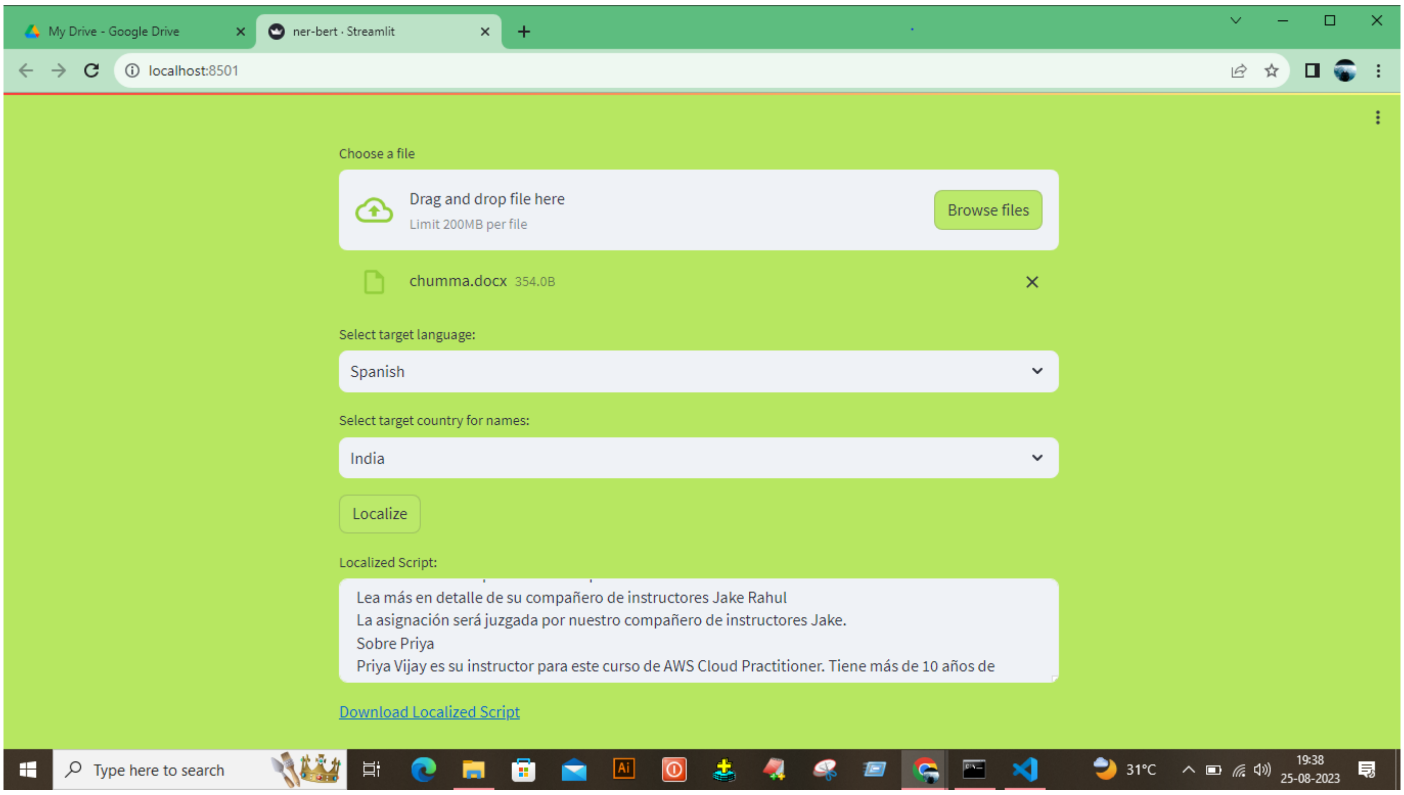
Browse files

if file not uploaded, PASTE YOUR SCRIPT HERE

Select target language:
Spanish

Select target country for names:
India

Localize



Choose a file

Drag and drop file here
Limit 200MB per file

Browse files

chumma.docx 354.0B

Select target language:
Spanish

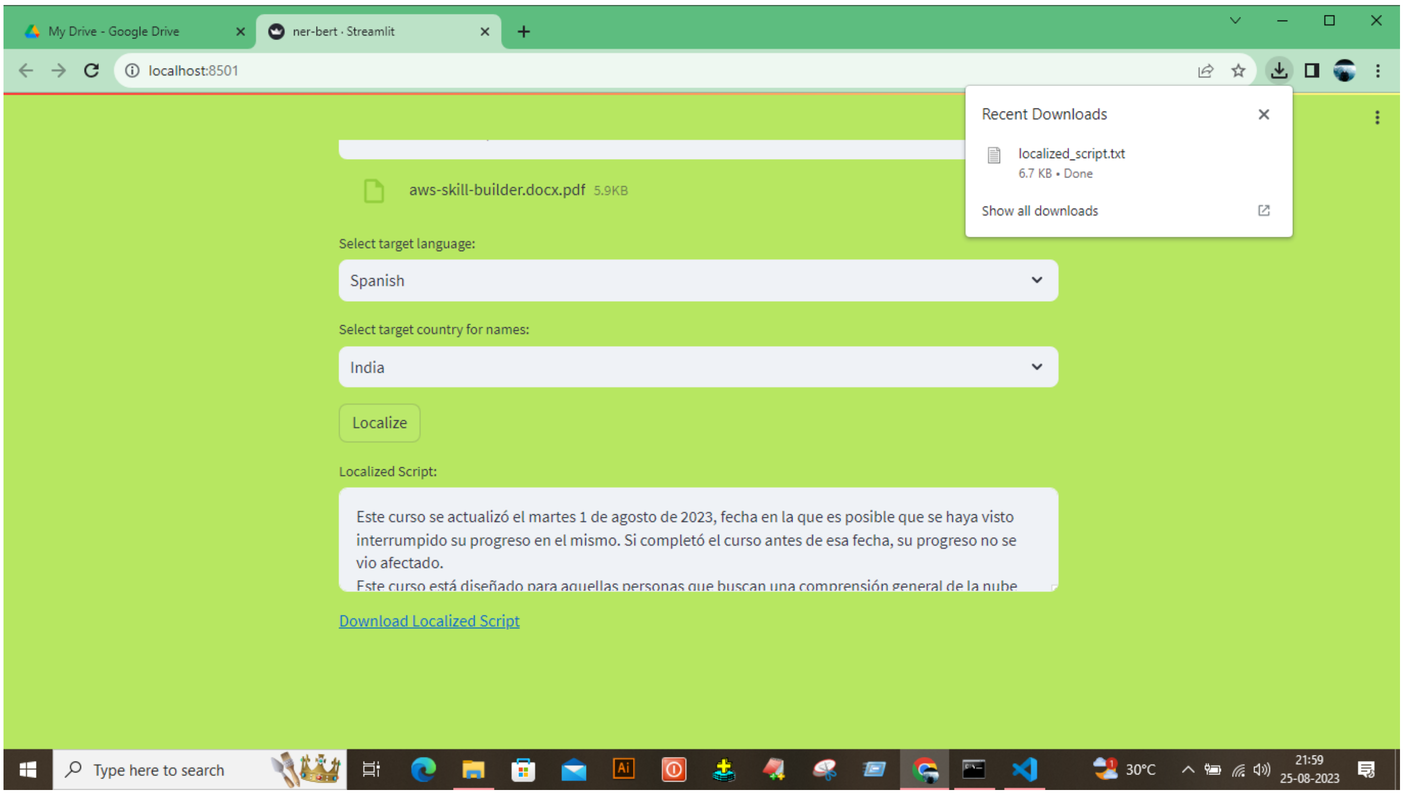
Select target country for names:
India

Localize

Localized Script:

Lea más en detalle de su compañero de instructores Jake Rahul
La asignación será juzgada por nuestro compañero de instructores Jake.
Sobre Priya
Priya Vijay es su instructor para este curso de AWS Cloud Practitioner. Tiene más de 10 años de

Download Localized Script



aws-skill-builder.docx.pdf 5.9KB

Select target language:

Spanish

Select target country for names:

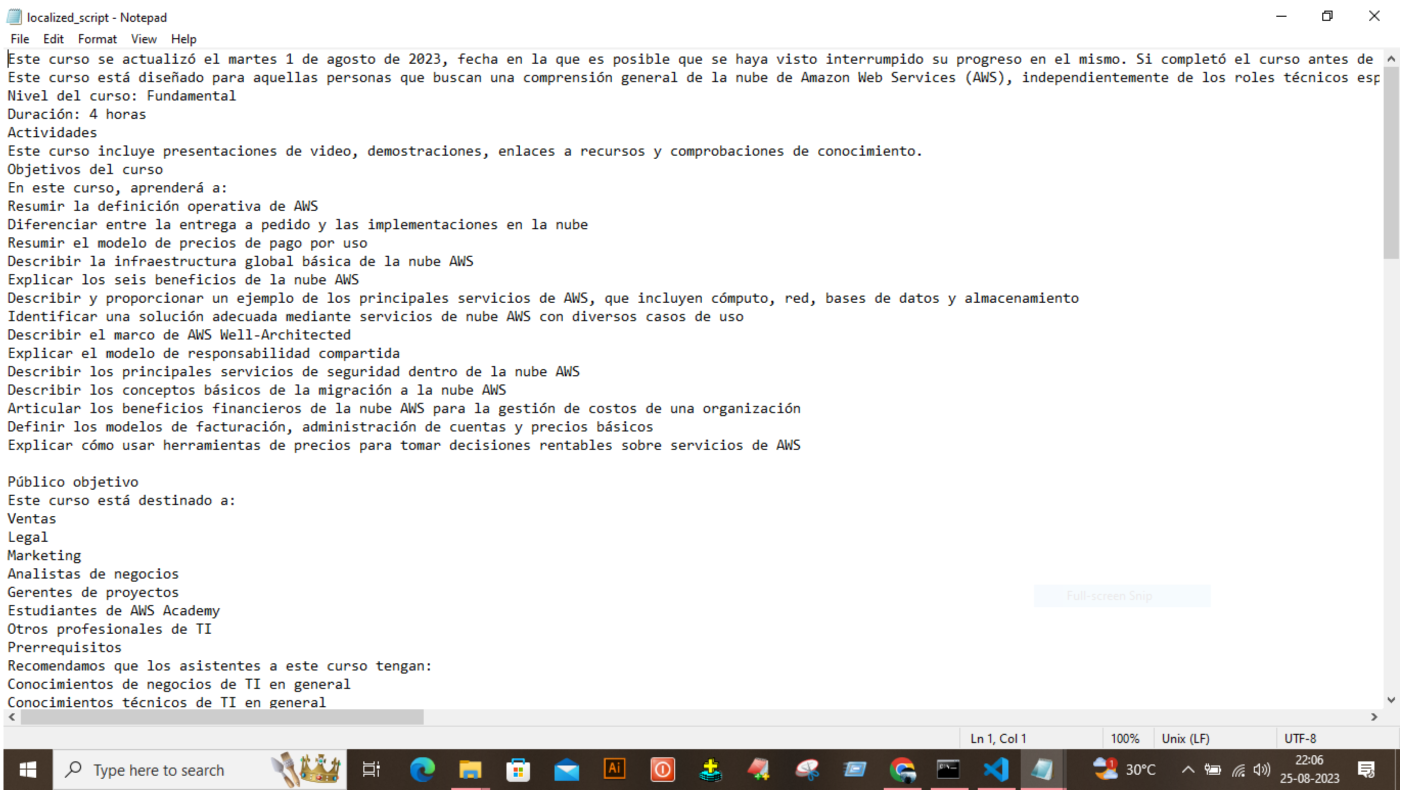
India

Localize

Localized Script:

Este curso se actualizó el martes 1 de agosto de 2023, fecha en la que es posible que se haya visto interrumpido su progreso en el mismo. Si completó el curso antes de esa fecha, su progreso no se vio afectado.
Este curso está diseñado para aquellas personas que buscan una comprensión general de la nube

Download Localized Script



```
Este curso se actualizó el martes 1 de agosto de 2023, fecha en la que es posible que se haya visto interrumpido su progreso en el mismo. Si completó el curso antes de
Este curso está diseñado para aquellas personas que buscan una comprensión general de la nube de Amazon Web Services (AWS), independientemente de los roles técnicos esp
Nivel del curso: Fundamental
Duración: 4 horas
Actividades
Este curso incluye presentaciones de video, demostraciones, enlaces a recursos y comprobaciones de conocimiento.
Objetivos del curso
En este curso, aprenderá a:
Resumir la definición operativa de AWS
Diferenciar entre la entrega a pedido y las implementaciones en la nube
Resumir el modelo de precios de pago por uso
Describir la infraestructura global básica de la nube AWS
Explicar los seis beneficios de la nube AWS
Describir y proporcionar un ejemplo de los principales servicios de AWS, que incluyen cómputo, red, bases de datos y almacenamiento
Identificar una solución adecuada mediante servicios de nube AWS con diversos casos de uso
Describir el marco de AWS Well-Architected
Explicar el modelo de responsabilidad compartida
Describir los principales servicios de seguridad dentro de la nube AWS
Describir los conceptos básicos de la migración a la nube AWS
Articular los beneficios financieros de la nube AWS para la gestión de costos de una organización
Definir los modelos de facturación, administración de cuentas y precios básicos
Explicar cómo usar herramientas de precios para tomar decisiones rentables sobre servicios de AWS

Público objetivo
Este curso está destinado a:
Ventas
Legal
Marketing
Analistas de negocios
Gerentes de proyectos
Estudiantes de AWS Academy
Otros profesionales de TI
Prerrequisitos
Recomendamos que los asistentes a este curso tengan:
Conocimientos de negocios de TI en general
Conocimientos técnicos de TI en general
```

4.REFERENCE:

1. Streamlit documentation - https://docs.streamlit.io/

2. Openai documentation - https://platform.openai.com/docs/

3.Deeplearning.AI courses - https://www.deeplearning.ai/short-courses/

4. Bert-base-NER-doc - https://huggingface.co/dslim/bert-base-NER#bert-base-ner