

```
%pip install --q unstructured langchain
%pip install --q "unstructured[all-docs]"
```

```
Collecting unstructured
  Downloading unstructured-0.1.0-py3-none-any.whl (1.4 MB)
    100% |██████████| 1.4/1.4 MB 10.6 MB/s eta 0:00:00

Collecting langchain
  Downloading langchain-0.2.5-py3-none-any.whl (2.2 MB)
    100% |██████████| 2.2/2.2 MB 8.5 MB/s eta 0:00:00

Preparing metadata (setup.py) ... done
  3.4/3.4 MB 30.5 MB/s eta 0:00:00
  41.0/41.0 kB 2.7 MB/s eta 0:00:00
  321.8/321.8 kB 17.1 MB/s eta 0:00:00
  127.4/127.4 kB 7.5 MB/s eta 0:00:00
  145.0/145.0 kB 5.4 MB/s eta 0:00:00
  49.2/49.2 kB 656.5 kB/s eta 0:00:00
  80.8/80.8 kB 4.0 MB/s eta 0:00:00
  75.6/75.6 kB 3.3 MB/s eta 0:00:00
  290.4/290.4 kB 19.6 MB/s eta 0:00:00
  54.5/54.5 kB 5.3 MB/s eta 0:00:00
  77.9/77.9 kB 6.8 MB/s eta 0:00:00
  58.3/58.3 kB 5.4 MB/s eta 0:00:00

Building wheel for langdetect (setup.py) ... done
  15.9/15.9 MB 60.4 MB/s eta 0:00:00
  2.4/2.4 MB 37.9 MB/s eta 0:00:00
  5.6/5.6 MB 70.9 MB/s eta 0:00:00
  56.4/56.4 kB 4.7 MB/s eta 0:00:00
  471.6/471.6 kB 32.8 MB/s eta 0:00:00
  112.5/112.5 kB 10.0 MB/s eta 0:00:00
  459.6/459.6 kB 33.1 MB/s eta 0:00:00
  244.3/244.3 kB 19.7 MB/s eta 0:00:00
  7.5/7.5 MB 71.5 MB/s eta 0:00:00
  19.2/19.2 MB 50.1 MB/s eta 0:00:00
  6.8/6.8 MB 68.8 MB/s eta 0:00:00
  2.3/2.3 MB 14.6 MB/s eta 0:00:00
  159.9/159.9 kB 15.3 MB/s eta 0:00:00
  79.5/79.5 kB 7.2 MB/s eta 0:00:00
  4.5/4.5 MB 48.0 MB/s eta 0:00:00
  114.6/114.6 kB 11.4 MB/s eta 0:00:00
  117.0/117.0 kB 10.2 MB/s eta 0:00:00

Preparing metadata (setup.py) ... done
  46.0/46.0 kB 4.0 MB/s eta 0:00:00
  21.3/21.3 MB 28.2 MB/s eta 0:00:00
  42.2/42.2 kB 3.6 MB/s eta 0:00:00

Preparing metadata (setup.py) ... done
  57.9/57.9 kB 5.4 MB/s eta 0:00:00
  86.8/86.8 kB 8.9 MB/s eta 0:00:00
  2.8/2.8 MB 75.6 MB/s eta 0:00:00
```

Building wheel for antlr4-python3-runtime (setup.py) ... done

Building wheel for iopath (setup.py) ... done

ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts

imageio 2.31.6 requires pillow<10.1.0,>=8.3.2, but you have pillow 10.3.0 which is incompatible.

```
!pip install langchain-community
```

```
Collecting langchain-community
  Downloading langchain_community-0.2.5-py3-none-any.whl (2.2 MB)
    100% |██████████| 2.2/2.2 MB 8.5 MB/s eta 0:00:00
```

```

Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (2.0.31)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (3.9.5)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (0.6.7)
Requirement already satisfied: langchain<0.3.0,>=0.2.5 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (0.2.5)
Requirement already satisfied: langchain-core<0.3.0,>=0.2.7 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (0.2.9)
Requirement already satisfied: langsmith<0.2.0,>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (0.1.82)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (1.25.2)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from langchain-community) (8.4.1)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (1.9.4)
Requirement already satisfied: asyncio-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (4.0.3)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in /usr/local/lib/python3.10/dist-packages (from dataclasses-json<0.7,>=0.5.7->langchain-community) (3.21.3)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from dataclasses-json<0.7,>=0.5.7->langchain-community) (0.9.0)
Requirement already satisfied: langchain-text-splitters<0.3.0,>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from langchain<0.3.0,>=0.2.5->langchain-community) (0.2.1)
Requirement already satisfied: pydantic<3,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain<0.3.0,>=0.2.5->langchain-community) (2.7.4)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.7->langchain-community) (1.33)
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.7->langchain-community) (24.1)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3.10/dist-packages (from langsmith<0.2.0,>=0.1.0->langchain-community) (3.10.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain-community) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain-community) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain-community) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain-community) (2024.6.2)
Requirement already satisfied: typing-extensions>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->langchain-community) (4.12.2)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->langchain-community) (3.0.3)
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.10/dist-packages (from jsonpatch<2.0,>=1.33->langchain-core<0.3.0,>=0.2.7->langchain-community) (3.
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1->langchain<0.3.0,>=0.2.5->langchain-community) (0.7.0)
Requirement already satisfied: pydantic-core==2.18.4 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1->langchain<0.3.0,>=0.2.5->langchain-community) (2.18.4)
Requirement already satisfied: mypy-extensions>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from typing-inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7->langchain-comm
Installing collected packages: langchain-community
Successfully installed langchain-community-0.2.5

```

```

from langchain_community.document_loaders import UnstructuredPDFLoader
from langchain_community.document_loaders import OnlinePDFLoader

```

```
# Instalación de PyPDF2
```

```
!pip install PyPDF2
```

```
# Importar las bibliotecas necesarias
import PyPDF2
```

↳ Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.10/dist-packages (3.0.1)

```
# Ruta local del archivo PDF
local_path = "/content/00f6e6_ManualMEGAMIG500Spanish.pdf"

# Función para extraer texto de un PDF usando UnstructuredPDFLoader y PyPDF2
def extract_text_from_pdf(pdf_path):
    loader = UnstructuredPDFLoader(file_path=pdf_path)
    pdf_obj = loader.load()

    pdf_reader = PyPDF2.PdfReader(open(pdf_path, "rb"))
    text = ''

    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num] # Acceso a las páginas usando un índice
        text += page.extract_text() # Uso del método extract_text()

    return text

# Función para formatear el texto en párrafos
def format_text_into_paragraphs(text):
    paragraphs = text.split('\n\n')
    formatted_text = '\n\n'.join(paragraph.strip() for paragraph in paragraphs if paragraph.strip())
    return formatted_text

# Función para guardar el texto en un archivo
def save_text_to_file(text, txt_path):
    with open(txt_path, 'w', encoding='utf-8') as file:
        file.write(text)

# Ruta de salida para el archivo .txt combinado
output_path = "/content/TextoconvertidoMEGAMIG500.txt"

# Extraer texto del PDF
pdf_text = extract_text_from_pdf(local_path)

# Formatear el texto en párrafos
formatted_text = format_text_into_paragraphs(pdf_text)

# Guardar el texto formateado en un archivo .txt
save_text_to_file(formatted_text, output_path)

print(f'Texto extraído y guardado en {output_path}')
```

→ Texto extraído y guardado en /content/TextoconvertidoMEGAMIG500.txt

```
! pip install faiss-cpu==1.7.4 mistralai

→ Collecting faiss-cpu==1.7.4
  Downloading faiss_cpu-1.7.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.6 MB)
    17.6/17.6 MB 53.9 MB/s eta 0:00:00
Collecting mistralai
  Downloading mistralai-0.4.1-py3-none-any.whl (19 kB)
Requirement already satisfied: httpx<1,>=0.25 in /usr/local/lib/python3.10/dist-packages (from mistralai) (0.27.0)
Requirement already satisfied: orjson<3.11,>=3.9.10 in /usr/local/lib/python3.10/dist-packages (from mistralai) (3.10.5)
Requirement already satisfied: pydantic<3,>=2.5.2 in /usr/local/lib/python3.10/dist-packages (from mistralai) (2.7.4)
```

```
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.25->mistralai) (3.7.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.25->mistralai) (2024.6.2)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.25->mistralai) (1.0.5)
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.25->mistralai) (3.7)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.25->mistralai) (1.3.1)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from httpcore==1.*->httpx<1,>=0.25->mistralai) (0.14.0)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=2.5.2->mistralai) (0.7.0)
Requirement already satisfied: pydantic-core==2.18.4 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=2.5.2->mistralai) (2.18.4)
Requirement already satisfied: typing-extensions>=4.6.1 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=2.5.2->mistralai) (4.12.2)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio->httpx<1,>=0.25->mistralai) (1.2.1)
Installing collected packages: faiss-cpu, mistralai
Successfully installed faiss-cpu-1.7.4 mistralai-0.4.1
```

```
from mistralai.client import MistralClient
from mistralai.models.chat_completion import ChatMessage
import requests
import numpy as np
import faiss
import os
from getpass import getpass

api_key= getpass("Type your API Key")
client = MistralClient(api_key=api_key)
```

⤵ Type your API Key.....

```
len(formatted_text)
⤵ 33176

text=formatted_text

chunk_size = 2048
chunks = [text[i:i + chunk_size] for i in range(0, len(text), chunk_size)]

def get_text_embedding(input):
    embeddings_batch_response = client.embeddings(
        model="mistral-embed",
        input=input
    )
    return embeddings_batch_response.data[0].embedding
```

```
text_embeddings = np.array([get_text_embedding(chunk) for chunk in chunks])
```

```
text_embeddings.shape
```

⤵ (17, 1024)

```
text_embeddings
```

```
↳ array([[-0.00710678,  0.03114319,  0.06506348, ...,  0.00994873,
       0.00481796, -0.01983643],
       [-0.01570129,  0.02578735,  0.05477905, ...,  0.01004791,
       -0.0033474 , -0.02294922],
       [-0.02400208,  0.01657104,  0.04620361, ...,  0.00724792,
       -0.00640488, -0.02763367],
       ...,
       [-0.02757263,  0.02531433,  0.05285645, ..., -0.01248932,
       -0.00832367, -0.01673889],
       [-0.03582764,  0.02583313,  0.05380249, ...,  0.00488281,
       -0.00343323, -0.00863647],
       [-0.0138092 ,  0.03259277,  0.03988647, ...,  0.01076508,
       0.0012989 , -0.00162315]])
```

```
d = text_embeddings.shape[1]
index = faiss.IndexFlatL2(d)
index.add(text_embeddings)
```

```
question = "¿Qué se debe hacer antes de soldar?"
question_embeddings = np.array([get_text_embedding(question)])
question_embeddings.shape
```

```
↳ (1, 1024)
```

```
question_embeddings
```

```
↳ array([[-0.04623413,  0.02729797,  0.05236816, ..., -0.00793457,
       -0.02848816, -0.00795746]])
```

```
D, I = index.search(question_embeddings, k=2)
print(I)
```

```
↳ [[3 9]]
```

```
retrieved_chunk = [chunks[i] for i in I.tolist()[0]]
print(retrieved_chunk)
```

```
↳ [''` ello solo conseguirá acortar la vida \nútil de los componentes y del equipo en general. \n% Siempre verifique que el cable de soldar no se encuentre maltratado. \n \n ``'
```

```
prompt = f"""
La información de contexto está a continuación..
-----
```

```
{retrieved_chunk}
```

```
-----
```

```
Dada la información del contexto y no el conocimiento previo, responda la consulta.
```

```
Consulta: {question}
```

```
Respuesta:
```

```
"""
```

```
def run_mistral(user_message, model="mistral-large-latest"):
    messages = [
        ChatMessage(role="user", content=user_message)
    ]
    chat_response = client.chat(
        model=model,
        messages=messages
    )
    return (chat_response.choices[0].message.content)
```

```
# Ejecuta la función run_mistral con el prompt generado
response = run_mistral(prompt)
```

```
# La respuesta generada por el modelo se almacena en la variable 'response'
print(response)
```

☞ Antes de soldar, se deben tomar varias medidas de seguridad y realizar un chequeo detallado. Aquí hay algunos pasos clave:

1. Verifique que no exista riesgo potencial de caída de cualquier objeto extraño, tanto para el operador como para la máquina.
2. Asegúrese de que el polvo, ácido o material corrosivo en el ambiente del lugar de trabajo y medio ambiente no sobrepasen los parámetros exigidos.
3. El equipo de soldar debe instalarse protegido del sol, la lluvia, humedad excesiva o temperaturas por debajo de los -10° C o por sobre los 40° C.
4. Deben existir al menos 50 cms. libres alrededor del equipo de soldar para asegurar una adecuada ventilación.
5. No debe introducir piezas extrañas al equipo de soldar.
6. No deben existir vibraciones excesivas en el área alrededor del lugar de trabajo de la máquina.
7. Elija un área sin interferencias electromagnéticas.
8. Asegúrese de instalar un interruptor automático de protección en la instalación eléctrica de acuerdo a las especificaciones indicadas por el manual de servicio del equipo
9. El equipo de soldar debe instalarse horizontalmente, en una superficie plana. Si la inclinación fuera superior a 15° se le debe adicionar elementos anti vuelco para evita
10. Verifique que la conexión de tierra de la toma eléctrica del equipo de soldar esté conectada a tierra en forma correcta.
11. Verifique que los cables de salida no estén en corto circuito.
12. Verifique que los cables de salida y alimentación estén en buenas condiciones, sin daños o alteraciones y/o fuera de estándar.
13. Los soldadores deben siempre utilizar los elementos de protección personal adecuados para prevenir daño por arco y radiación térmica.
14. Es necesario colocar cortinas protectoras al rededor del lugar de trabajo para prevenir accidentes a terceros por efecto del arco eléctrico producido en el proceso de so
15. En los alrededores del sitio de trabajo no debe haber productos inflamables o explosivos.
16. Cada conexión en el equipo de soldar debe realizarse de manera correcta y segura.

Después de estas verificaciones y medidas de seguridad, puede encender la máquina y regular la corriente de soldadura al valor requerido mediante la perilla de ajuste de cor

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

☞ [nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
True

```
!pip install rouge-score
from rouge_score import rouge_scorer
```

☞ Collecting rouge-score
 Downloading rouge_score-0.1.2.tar.gz (17 kB)
 Preparing metadata (setup.py) ... done
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from rouge-score) (1.4.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (from rouge-score) (3.8.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from rouge-score) (1.25.2)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from rouge-score) (1.16.0)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk->rouge-score) (8.1.7)

```
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk->rouge-score) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk->rouge-score) (2024.5.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk->rouge-score) (4.66.4)
Building wheels for collected packages: rouge-score
  Building wheel for rouge-score (setup.py) ... done
  Created wheel for rouge-score: filename=rouge_score-0.1.2-py3-none-any.whl size=24933 sha256=0916a48059f6d71f506238f3c709863abb632253914d78cf8ece73352f070966
  Stored in directory: /root/.cache/pip/wheels/5f/dd/89/461065a73be61a532ff8599a28e9beef17985c9e9c31e541b4
Successfully built rouge-score
Installing collected packages: rouge-score
Successfully installed rouge-score-0.1.2
```

```
!pip install bert-score
from bert_score import score
```

Collecting bert-score

 Downloading bert_score-0.3.13-py3-none-any.whl (61 kB)

 61.1/61.1 kB 543.3 kB/s eta 0:00:00

```
Requirement already satisfied: torch>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from bert-score) (2.3.0+cu121)
Requirement already satisfied: pandas>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from bert-score) (2.0.3)
Requirement already satisfied: transformers>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from bert-score) (4.41.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from bert-score) (1.25.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from bert-score) (2.31.0)
Requirement already satisfied: tqdm>=4.31.1 in /usr/local/lib/python3.10/dist-packages (from bert-score) (4.66.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from bert-score) (3.7.1)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from bert-score) (24.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.1->bert-score) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.1->bert-score) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.1->bert-score) (2024.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (3.15.3)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (1.12.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (2023.6.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (12.1.105)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (12.1.105)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (12.1.105)
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (8.9.2.26)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (12.1.3.1)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (11.0.2.54)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (10.3.2.106)
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (11.4.5.107)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (12.1.0.106)
Requirement already satisfied: nvidia-nccl-cu12==2.20.5 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (2.20.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (12.1.105)
Requirement already satisfied: triton==2.3.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.0.0->bert-score) (2.3.0)
Requirement already satisfied: nvidia-nvjitlink-cu12 in /usr/local/lib/python3.10/dist-packages (from nvidia-cusolver-cu12==11.4.5.107->torch>=1.0.0->bert-score) (12.5.40)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-score) (0.23.4)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-score) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-score) (2024.5.15)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-score) (0.19.1)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers>=3.0.0->bert-score) (0.4.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->bert-score) (3.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->bert-score) (3.3.2)
```

```
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->bert-score) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->bert-score) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->bert-score) (2024.6.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0.1->bert-score) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.0.0->bert-score) (2.1.5)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.0.0->bert-score) (1.3.0)
Installing collected packages: bert-score
Successfully installed bert-score-0.3.13
```

```
# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = """Antes de soldar verifique el estado de las conexiones.
Revise que la conexión de tierra del enchufe esté correcta.
Utilice ropa y herramientas apropiadas para evitar dañar la vista y la piel.
Cuando se este soldando se debe usar la máscara de soldar cubriendo toda la cabeza, sólo se puede realizar la observación visual del arco eléctrico a través del visor de la máscara.
Evite la sobrecarga de su equipo revisando previamente el ciclo de trabajo de este. Tome la precaución de que la manguera de gas nunca se encuentre presionada o doblada. Siempre Asegúrese de que no exista riesgo potencial, tanto para el operador como para la máquina, de caída de cualquier objeto extraño.
El polvo, ácido o material corrosivo en el ambiente del lugar de trabajo y medio ambiente corrosivo no deben sobrepasar los parámetros exigidos (exceptuándose los provocados a causa de la actividad).
El equipo de soldar debe instalarse protegido del sol, la lluvia, humedad excesiva o temperaturas por bajo los -10° C o por sobre los 40° C.
Deben existir al menos 50 cms. libres alrededor del equipo de soldar para asegurar una adecuada ventilación.
No debe introducir piezas extrañas al equipo de soldar.
No deben existir vibraciones excesivas en el área alrededor del lugar de trabajo de la máquina.
Elija un área sin interferencias electromagnéticas.
Asegúrese de instalar un interruptor automático de protección en la instalación eléctrica de acuerdo a las especificaciones indicadas por el manual de servicio del equipo.
El equipo de soldar debe instalarse horizontalmente, en una superficie plana.
Si la inclinación fuera superior a 15° se le debe adicionar elementos anti vuelco para evitar la inclinación del equipo.
No se recomienda conectar su equipo de soldar a un grupo generador.""" # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scoring.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}")
print(f"ROUGE-2 score: {rouge_scores['rouge2']}")
print(f"ROUGE-L score: {rouge_scores['rougeL']}")
print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")
```

```
↳ /usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:  
  The secret `HF_TOKEN` does not exist in your Colab secrets.  
  To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens)  
  You will be able to reuse this secret in all of your notebooks.  
  Please note that authentication is recommended but still optional to access public models or datasets.  
    warnings.warn(  
tokenizer_config.json: 100%                                         49.0/49.0 [00:00<00:00, 2.23kB/s]  
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is depr  
  warnings.warn(  
config.json: 100%                                         625/625 [00:00<00:00, 8.09kB/s]  
vocab.txt: 100%                                         996k/996k [00:00<00:00, 4.63MB/s]  
tokenizer.json: 100%                                         1.96M/1.96M [00:00<00:00, 18.3MB/s]  
model.safetensors: 100%                                         714M/714M [00:12<00:00, 94.7MB/s]  
calculating scores...  
computing bert embedding.  
100%                                         1/1 [00:09<00:00, 9.06s/it]  
computing greedy matching.  
100%                                         1/1 [00:00<00:00, 12.87it/s]  
done in 9.16 seconds, 0.11 sentences/sec  
ROUGE-1 score: Score(precision=0.6190476190476191, recall=0.8024691358024691, fmeasure=0.6989247311827957)  
ROUGE-2 score: Score(precision=0.4606205250596659, recall=0.5975232198142415, fmeasure=0.5202156334231807)  
ROUGE-L score: Score(precision=0.44761904761904764, recall=0.5802469135802469, fmeasure=0.5053763440860215)  
BERTScore - Precision: 0.8191155791282654, Recall: 0.8349132537841797, F1: 0.8269389271736145
```

```
question = "¿Qué ocurre si sucede Shock Eléctrico?"  
question_embeddings = np.array([get_text_embedding(question)])  
question_embeddings.shape
```

```
↳ (1, 1024)
```

```
D, I = index.search(question_embeddings, k=2)  
print(I)
```

```
↳ [[4 3]]
```

```
retrieved_chunk = [chunks[i] for i in I.tolist()[0]]  
print(retrieved_chunk)  
  
↳ ['lética antes de abrir la carcaza. \n; Ante cualquier problema técnico que presente el equipo de soldar, recurra al servicio técnico \nautorizado más cercano. \n \nSe deb
```

```

prompt = f"""
La información de contexto está a continuación..
-----
{retrieved_chunk}
-----
Dada la información del contexto y no el conocimiento previo, responda la consulta.
Consulta: {question}
Respuesta:
"""

# Ejecuta la función run_mistral con el prompt generado
response = run_mistral(prompt)

# La respuesta generada por el modelo se almacena en la variable 'response'
print(response)

➡️ ilios equipado para socorrer a posibles víctimas de un shock eléctrico y tratar posibles quemaduras a la piel y/o ojos causadas por la exposición directa a la luz y calor emi

# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = """Si la persona accidentada está inconsciente y se sospecha un shock eléctrico, tenga la precaución de no tocarla si ha quedado en contacto con algún cuerpo posiblemente energizado.
Corte el suministro eléctrico que alimenta el equipo y recurra a los cuidados de primeros auxilios.
Para alejar los cables y/o partes energizadas de la víctima, se recomienda utilizar trozos de madera bien seca, como una escoba o cualquier otro material aislante.""" # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}") 
print(f"ROUGE-2 score: {rouge_scores['rouge2']}") 
print(f"ROUGE-L score: {rouge_scores['rougeL']}") 
print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")

➡️ calculating scores...
computing bert embedding.
100%                                         1/1 [00:01<00:00, 1.62s/it]
computing greedy matching.
100%                                         1/1 [00:00<00:00, 10.96it/s]
done in 1.73 seconds, 0.58 sentences/sec
ROUGE-1 score: Score(precision=0.4755244755244755, recall=0.8947368421052632, fmeasure=0.6210045662100456)
ROUGE-2 score: Score(precision=0.39436619718309857, recall=0.74666666666666667, fmeasure=0.5161290322580645)
ROUGE-L score: Score(precision=0.4195804195804196, recall=0.7894736842105263, fmeasure=0.547945205479452)
BERTScore - Precision: 0.8026891350746155, Recall: 0.8870231509208679, F1: 0.8427515625953674

question = ";Cuáles son los requerimientos para la fuente de poder?"
https://colab.research.google.com/drive/1WiUYhg8cGWczcrevthfvcJ_RkpOrPqO#scrollTo=a5l10nbIqOL4&printMode=true

```

```

    -----  

    question_embeddings = np.array([get_text_embedding(question)])  

question_embeddings.shape  

D, I = index.search(question_embeddings, k=2)  

retrieved_chunk = [chunks[i] for i in I.tolist()[0]]  

prompt = f"""  

La información de contexto está a continuación..  

-----  

{retrieved_chunk}  

-----  

Dada la información del contexto y no el conocimiento previo, responda la consulta.  

Consulta: {question}  

Respuesta:  

"""  

# Ejecuta la función run_mistral con el prompt generado  

response = run_mistral(prompt)  

  

# La respuesta generada por el modelo se almacena en la variable 'response'  

print(response)

```

→ Los requerimientos para la fuente de poder según la información proporcionada son:

- a) El oscilograma de voltaje debe mostrar una onda de seno, y la oscilación de la frecuencia no debe exceder $\pm 1\%$ del valor indicado.
- b) La oscilación del voltaje no debe superar $\pm 10\%$ del valor indicado.
- c) El desbalance de la alimentación trifásica no debe exceder el 5%.

Además, los datos técnicos para el equipo de soldar MEGAMIG 500 mencionan que el voltaje de entrada es de 3PH - 380 V, la frecuencia es de 50 Hz, y la corriente de entrada es

```

# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = """El oscilograma de voltaje debe mostrar una onda de seno, la oscilación de la frecuencia no  

debe exceder  $\pm 1\%$  del valor indicado. La oscilación del voltaje no debe superar  $\pm 10\%$  del valor indicado.  

El desbalance de la alimentación trifásica no debe exceder el 5%.""" # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}")  

print(f"ROUGE-2 score: {rouge_scores['rouge2']}")  

print(f"ROUGE-L score: {rouge_scores['rougeL']}")  

print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")

```

```
→ calculating scores...
computing bert embedding.
100% 1/1 [00:01<00:00, 1.49s/it]
computing greedy matching.
100% 1/1 [00:00<00:00, 18.32it/s]

done in 1.55 seconds, 0.64 sentences/sec
ROUGE-1 score: Score(precision=0.42105263157894735, recall=1.0, fmeasure=0.5925925925925926)
ROUGE-2 score: Score(precision=0.3893805309734513, recall=0.9361702127659575, fmeasure=0.55)
ROUGE-L score: Score(precision=0.42105263157894735, recall=1.0, fmeasure=0.5925925925925926)
BERTScore - Precision: 0.7853518724441528, Recall: 0.9210900664329529, F1: 0.847822368144989
```

```
question = "¿Cuáles son las principales tareas de mantenimiento?"
question_embeddings = np.array([get_text_embedding(question)])
question_embeddings.shape
D, I = index.search(question_embeddings, k=2)
retrieved_chunk = [chunks[i] for i in I.tolist()[0]]
prompt = f"""
La información de contexto está a continuación..
-----
{retrieved_chunk}
-----
Dada la información del contexto y no el conocimiento previo, responda la consulta.
Consulta: {question}
Respuesta:
"""

# Ejecuta la función run_mistral con el prompt generado
response = run_mistral(prompt)

# La respuesta generada por el modelo se almacena en la variable 'response'
print(response)
```

→ Las principales tareas de mantenimiento para el equipo de soldar incluyen:

1. Remoción de Polvo: Asegúrate de que la máquina no esté conectada a una fuente de poder antes de proceder a la remoción de polvo.
2. Almacenamiento seguro: Si vas a dejar tu equipo por un largo periodo de inactividad, hazlo en un ambiente seco, limpio y con buena ventilación.
3. Limpieza periódica: A través de personal especializado, se debe realizar limpieza por medio de aire comprimido seco para limpiar el interior del equipo de soldar. También
4. Mantenimiento del cable de poder: Es necesario chequear periódicamente el cable de alimentación del equipo de soldar.
5. Revisión de Cables de soldar: Al momento de utilizar el equipo de soldar, el operador debe revisar que los cables de soldar no estén desconectados y no presenten cortes e
6. Revisión del circuito de gas: Revise regularmente que el circuito de gas no presente fugas y se mantenga completamente sellado.
7. Limpieza de la boquilla: Limpie periódicamente la boquilla de escoria y suciedad.
8. Uso de alambre Mig de calidad: Utilice alambre Mig de calidad, que cuente con certificaciones.
9. Reemplazo de rodillos: Reemplace los rodillos si se encuentran gastados o deteriorados.

Estas tareas deben realizarse con regularidad para asegurar el correcto funcionamiento y prolongar la vida útil del equipo de soldar.

```

# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = """Remoción de Polvo .Debe asegurarse que la máquina no esté conectada a una fuente de
poder antes de proceder a la remoción de polvo.
Asegurece que al dejar su equipo por un largo periodo de inactividad, lo haga en un ambiente seco, limpio y con buena ventilación.
Limpieza periódica. A través de personal especializado, se debe realizar limpieza por medio de aire comprimido seco para limpiar el interior del equipo de soldar. También se
debe revisar si existen componentes o cables sueltos, si los hubiera, proceda a la reparación o reemplazo correspondiente. Generalmente cuando no existe polvo en demasía la limp
se debe realizar periódicamente, y cuando existe mucha acumulación de polvo la limpieza se debe realizar con mayor frecuencia.Mantenga el cable de poder en buen estado Es nesca
de alimentación del equipo de soldar, sin embargo, es aconsejable revisar cada vez que la máquina sea utilizada de forma portátil.
Revise Cables de soldar. Al momento de utilizar el equipo de soldar, el operador debe revisar que los cables de soldar no estén desconectados, no presenten cortes en el
aislamiento, etc. En caso de que se presente alguno de estos problemas se debe proceder al cambio o reparación correspondiente.
Revise regularmente que el circuito de gas no presente fugas y se mantenga completamente sellado. Verifique ademas que el ventilador y el motor alimentador estén funcionando
correctamente sin sonidos anormales ni cables sueltos. Limpie periódicamente la boquilla de escoria y suciedad.
Utilice alambre Mig de calidad, que cuente con certificaciones. Reemplace los rodillos si se encuentran gastados o deteriorados. Apriete los rodillos contra
el alambre sin sobrepresionar para asegurar un buen desplazamiento de éste.""" # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougel'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}")"
print(f"ROUGE-2 score: {rouge_scores['rouge2']}")"
print(f"ROUGE-L score: {rouge_scores['rougel']}")"
print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")

➡️ calculating scores...
computing bert embedding.
100% 1/1 [00:08<00:00, 8.62s/it]
computing greedy matching.
100% 1/1 [00:00<00:00, 9.84it/s]

done in 8.75 seconds, 0.11 sentences/sec
ROUGE-1 score: Score(precision=0.7874015748031497, recall=0.6734006734006734, fmeasure=0.7259528130671508)
ROUGE-2 score: Score(precision=0.6521739130434783, recall=0.5574324324324325, fmeasure=0.6010928961748634)
ROUGE-L score: Score(precision=0.7165354330708661, recall=0.6127946127946128, fmeasure=0.6606170598911071)
BERTScore - Precision: 0.7953457832336426, Recall: 0.7899434566497803, F1: 0.792635440826416

```

```

question = "¿Antes de armar o desarmar la pistola que se debe hacer?"
question_embeddings = np.array([get_text_embedding(question)])
question_embeddings.shape
D, I = index.search(question_embeddings, k=2)
retrieved_chunk = [chunks[i] for i in I.tolist()[0]]
prompt = f"""
La información de contexto está a continuación..
-----
{retrieved_chunk}
-----
Dada la información del contexto y no el conocimiento previo, responda la consulta.
Consulta: {question}
Respuesta:
"""
# Ejecuta la función run_mistral con el prompt generado
response = run_mistral(prompt)

# La respuesta generada por el modelo se almacena en la variable 'response'
print(response)

→ ola, se debe desconectar el equipo de soldar de la fuente de energía eléctrica. Esto es para garantizar la seguridad del operador y evitar daños a los componentes y al equipo


# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = "Antes de armar o desarmar la pistola se la debe desconectar de la energía eléctrica." # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}")
print(f"ROUGE-2 score: {rouge_scores['rouge2']}")
print(f"ROUGE-L score: {rouge_scores['rougeL']}")
print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")

→ calculating scores...
computing bert embedding.
100% 1/1 [00:00<00:00, 1.54it/s]
computing greedy matching.
100% 1/1 [00:00<00:00, 18.96it/s]
done in 0.73 seconds, 1.37 sentences/sec
ROUGE-1 score: Score(precision=0.40476190476190477, recall=1.0, fmeasure=0.576271186440678)
ROUGE-2 score: Score(precision=0.2926829268292683, recall=0.75, fmeasure=0.4210526315789473)
ROUGE-L score: Score(precision=0.38095238095238093, recall=0.9411764705882353, fmeasure=0.5423728813559321)
BERTScore - Precision: 0.7762361764907837, Recall: 0.923952043056488, F1: 0.8436772227287292

question = "¿Cuándo pueden resultar peligrosas las operaciones de soldadura?"

```

```

question_embeddings = np.array([get_text_embedding(question)])
question_embeddings.shape
D, I = index.search(question_embeddings, k=2)
retrieved_chunk = [chunks[i] for i in I.tolist()[0]]
prompt = f"""
La información de contexto está a continuación..
-----
{retrieved_chunk}
-----
Dada la información del contexto y no el conocimiento previo, responda la consulta.
Consulta: {question}
Respuesta:
"""

# Ejecuta la función run_mistral con el prompt generado
response = run_mistral(prompt)

# La respuesta generada por el modelo se almacena en la variable 'response'
print(response)

```

→ Las operaciones de soldadura pueden resultar peligrosas en varias situaciones según la información proporcionada:

1. Cuando no se verifica el estado de las conexiones y la conexión de tierra del enchufe no está correcta.
2. Cuando no se trabaja en lugares bien ventilados o con extracción forzada, lo que puede causar la acumulación de humos y gases dañinos para la salud.
3. Cuando no se protege la zona de arco eléctrico mediante el uso de cortinas protectoras adecuadas, lo que puede afectar la vista de las personas cercanas.
4. Cuando personal no capacitado intenta mover o realizar ajustes al equipo de soldar.
5. Cuando personas con marcapasos cardíacos u otros artefactos sensibles se acercan al equipo de soldar en plena operación, debido a la inducción electromagnética producida.
6. Cuando se utiliza el equipo de soldar para descongelar tuberías o sin los paneles, lo que puede ser peligroso para el operador y dañar seriamente el funcionamiento del equipo.
7. Cuando se sobrecarga el equipo de soldadura sin revisar previamente su ciclo de trabajo.
8. Cuando la manguera de gas se encuentra presionada, doblada o con un radio de giro menor de 150 mm, lo que puede causar daño en su interior.
9. Cuando la pureza del CO₂ es menor que 99,5 % o su contenido de agua es mayor que 0,005 %.
10. Cuando se utiliza la máquina por sobre su ciclo de trabajo o se provoca que se active el termostato de seguridad.

```

# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = """Si las normas de seguridad y de utilización no se observan atentamente, las operaciones de soldadura pueden resultar peligrosas no solo para el operador, sino incluso para las personas que se encuentren próximas al área de trabajo.""" # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}")
print(f"ROUGE-2 score: {rouge_scores['rouge2']}")
print(f"ROUGE-L score: {rouge_scores['rougeL']}")
print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")

```

```

➡️ calculating scores...
computing bert embedding.
100% 1/1 [00:03<00:00, 3.09s/it]
computing greedy matching.
100% 1/1 [00:00<00:00, 15.84it/s]

done in 3.18 seconds, 0.31 sentences/sec
ROUGE-1 score: Score(precision=0.10344827586206896, recall=0.6923076923076923, fmeasure=0.18)
ROUGE-2 score: Score(precision=0.05, recall=0.34210526315789475, fmeasure=0.08724832214765102)
ROUGE-L score: Score(precision=0.06130268199233716, recall=0.41025641025641024, fmeasure=0.106666666666666666)
BERTScore - Precision: 0.6355978846549988, Recall: 0.7504872679710388, F1: 0.6882811188697815

question = "¿Cómo influye la velocidad del alambre?"
question_embeddings = np.array([get_text_embedding(question)])
question_embeddings.shape
D, I = index.search(question_embeddings, k=2)
retrieved_chunk = [chunks[i] for i in I.tolist()[0]]
prompt = f"""
La información de contexto está a continuación..
-----
{retrieved_chunk}
-----
Dada la información del contexto y no el conocimiento previo, responda la consulta.
Consulta: {question}
Respuesta:
"""
# Ejecuta la función run_mistral con el prompt generado
response = run_mistral(prompt)

# La respuesta generada por el modelo se almacena en la variable 'response'
print(response)

➡️ used para ajustar la corriente de soldadura y también para establecer el tiempo de Burn Back. Por lo tanto, la velocidad del alambre tiene un impacto significativo en la cali
◀ ▶

# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = """La velocidad del alambre influye directamente en la corriente de soldadura. A un mismo voltaje, a mayor velocidad del alambre mayor es la corriente de soldadura.""" # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}")
print(f"ROUGE-2 score: {rouge_scores['rouge2']}")
print(f"ROUGE-L score: {rouge_scores['rougeL']}")
print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")

```

```
→ calculating scores...
computing bert embedding.
100% 1/1 [00:00<00:00, 1.35it/s]
computing greedy matching.
100% 1/1 [00:00<00:00, 27.24it/s]
done in 0.79 seconds, 1.27 sentences/sec
ROUGE-1 score: Score(precision=0.2235294117647059, recall=0.7307692307692307, fmeasure=0.34234234234234234)
ROUGE-2 score: Score(precision=0.14285714285714285, recall=0.48, fmeasure=0.2201834862385321)
ROUGE-L score: Score(precision=0.2, recall=0.6538461538461539, fmeasure=0.3063063063063063)
BERTScore - Precision: 0.7516089081764221, Recall: 0.7950412034988403, F1: 0.7727152109146118
```

```
question = "¿Cómo se debe conectar el equipo de soldar y el sistema de gas?"
question_embeddings = np.array([get_text_embedding(question)])
question_embeddings.shape
D, I = index.search(question_embeddings, k=2)
retrieved_chunk = [chunks[i] for i in I.tolist()[0]]
prompt = f"""
La información de contexto está a continuación..
-----
{retrieved_chunk}
-----
Dada la información del contexto y no el conocimiento previo, responda la consulta.
Consulta: {question}
Respuesta:
"""
# Ejecuta la función run_mistral con el prompt generado
response = run_mistral(prompt)

# La respuesta generada por el modelo se almacena en la variable 'response'
print(response)
```

→ esté firmemente atornillado a su conexión para evitar fugas de gas. Luego, conecta y fija firmemente el cable de calefacción a la conexión del calefactor del panel trasero de

```

# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = """Asegúrese que el regulador de gas este firmemente atornillado a su conexión para evitar fugas de
gas. Luego conecte y fije firmemente el cable de calefacción a la conexión del calefactor del panel trasero
de la fuente de poder.
Conecte un extremo de la manguera de gas a la salida del regulador de presión y el otro extremo a la
entrada de gas del alimentador de alambre. """ # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}")  

print(f"ROUGE-2 score: {rouge_scores['rouge2']}")  

print(f"ROUGE-L score: {rouge_scores['rougeL']}")  

print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")

[?] calculating scores...  

computing bert embedding.  

100% 1/1 [00:00<00:00, 1.52it/s]  

computing greedy matching.  

100% 1/1 [00:00<00:00, 45.47it/s]  

done in 0.68 seconds, 1.46 sentences/sec  

ROUGE-1 score: Score(precision=0.8035714285714286, recall=0.625, fmeasure=0.703125)  

ROUGE-2 score: Score(precision=0.6909090909090909, recall=0.5352112676056338, fmeasure=0.6031746031746031)  

ROUGE-L score: Score(precision=0.6964285714285714, recall=0.5416666666666666, fmeasure=0.6093749999999999)  

BERTScore - Precision: 0.872178852558136, Recall: 0.8601152896881104, F1: 0.8661050796508789

question = "¿Cómo se debe realizar la instalación y conexión del alimentador de alambre SUPER TRACK?"  

question_embeddings = np.array([get_text_embedding(question)])  

question_embeddings.shape  

D, I = index.search(question_embeddings, k=2)  

retrieved_chunk = [chunks[i] for i in I.tolist()[0]]  

prompt = f"""  

La información de contexto está a continuación..  

-----  

{retrieved_chunk}  

-----  

Dada la información del contexto y no el conocimiento previo, responda la consulta.  

Consulta: {question}  

Respuesta:  

"""  

# Ejecuta la función run_mistral con el prompt generado
response = run_mistral(prompt)

# La respuesta generada por el modelo se almacena en la variable 'response'
print(response)

```

→ la instalación y conexión del alimentador de alambre SUPER TRACK, siga los siguientes pasos:

1 extremo de la manguera de gas a la salida del regulador de presión y el otro extremo a la entrada de gas del alimentador de alambre.

• el diámetro del alambre según el trabajo a ejecutar. En función de este diámetro, elija el carrete, flexible y boquilla adecuados.

Abra la cubierta del alimentador de alambre SUPER TRACK e instale el carrete en el eje del carrete.

Introduzca el alambre en el tubo de entrada del mecanismo de arrastre y alíñelo en la ranura del rodillo de empuje adecuado.

Centrarlo en la boquilla de salida y active el mecanismo de presión.

• pistola en el conector correspondiente del panel frontal y apriete firmemente.

1 extremo del cable de tierra al conector correspondiente en el panel frontal de la fuente de poder y atornille firmemente. Con el otro extremo, aprisione firmemente la pieza

```
# para comparar
generated_response = str(response) # La respuesta generada por el modelo
# La respuesta textual del manual técnico
reference_response = """Seleccione el diámetro del alambre según el trabajo a ejecutar. En función de este diámetro elija el
carrete, flexible y boquilla adecuados.
Abra la cubierta del alimentador de alambre SUPER TRACK e instale en el eje del carrete.
Introduzca el alambre en el tubo de entrada del mecanismo de arrastre y alíñelo en la ranura del
rodillo de empuje adecuado. Vuelva a centrarlo en la boquilla de salida y active el mecanismo de
presión. """ # La respuesta esperada

# Calcular ROUGE score
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougel'], use_stemmer=True)
rouge_scores = scorer.score(reference_response, generated_response)

# Calcular BERTScore
P, R, F1 = score([generated_response], [reference_response], lang="es", verbose=True)

# Imprime resultados
print(f"ROUGE-1 score: {rouge_scores['rouge1']}")  
print(f"ROUGE-2 score: {rouge_scores['rouge2']}")  
print(f"ROUGE-L score: {rouge_scores['rougel']}")  
print(f"BERTScore - Precision: {P.mean().item()}, Recall: {R.mean().item()}, F1: {F1.mean().item()}")
```

→ calculating scores...

computing bert embedding.

100% 1/1 [00:01<00:00, 1.39s/it]

computing greedy matching.

100% 1/1 [00:00<00:00, 34.52it/s]

done in 1.43 seconds, 0.70 sentences/sec

ROUGE-1 score: Score(precision=0.43646408839779005, recall=0.9875, fmeasure=0.6053639846743295)

ROUGE-2 score: Score(precision=0.40555555555555556, recall=0.9240506329113924, fmeasure=0.5637065637065638)

ROUGE-L score: Score(precision=0.43646408839779005, recall=0.9875, fmeasure=0.6053639846743295)

BERTScore - Precision: 0.786047101020813, Recall: 0.8880327939987183, F1: 0.833933413028717

```
question = "¿Qué sucede si el equipo de soldar es operado por largos períodos de tiempo?"
```

```
question_embeddings = np.array([get_text_embedding(question)])
```

```
question_embeddings.shape
```