```
!pip install -U sentence-transformers
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 1.4 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.6 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.7 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.4 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 7.0 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 93.4 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvr
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
  Attempting uninstall: sentence-transformers
    Found existing installation: sentence-transformers 4.1.0
    Uninstalling sentence-transformers-4.1.0:
      Successfully uninstalled sentence-transformers-4.1.0
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime
```

```
from sentence_transformers import SentenceTransformer
import pandas as pd

# Load the CSV file
df = pd.read_csv("/content/CDMX_Sentiment_Comparation.csv", encoding="latin1")

# Ensure consistent types
df["Number"] = df["Number"].astype(str)
df["respuesta"] = df["respuesta"].astype(str)

# Load the multilingual embedding model
model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")

# Filter real and generated responses
df_real = df[df["tipo"] == "real"]
df_gen = df[df["tipo"] == "generada"]

# Group responses by question number (concatenate all responses per question)
```

```
grouped_real = df_real.groupby("Number")["respuesta"].apply(lambda x: " ".join(x)).reset_index()
grouped_gen = df_gen.groupby("Number")["respuesta"].apply(lambda x: " ".join(x)).reset_index()

# Merge both groups on the question number
df_merged = pd.merge(grouped_real, grouped_gen, on="Number", suffixes=("_real", "_gen"))

# Generate embeddings
emb_real = model.encode(df_merged["respuesta_real"].tolist(), convert_to_tensor=True)
emb_gen = model.encode(df_merged["respuesta_gen"].tolist(), convert_to_tensor=True)
```

```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Filter to keep only question numbers present in both sets
valid_numbers = set(df_real["Number"]).intersection(set(df_gen["Number"]))
df_valid = df_merged[df_merged["Number"].isin(valid_numbers)].copy()

# Initialize list for similarity scores
similarities = []

# Compute cosine similarity for each valid question
for i, row in df_valid.iterrows():
    try:
        # Generate embeddings
        emb_r = model.encode(row["respuesta_real"], convert_to_tensor=True).unsqueeze(0)
        emb_g = model.encode(row["respuesta_gen"], convert_to_tensor=True).unsqueeze(0)

        # Compute cosine similarity
        sim = cosine_similarity(emb_r.cpu().numpy(), emb_g.cpu().numpy())[0][0]
    except Exception as e:
        sim = None

    similarities.append(sim)

# Add results to the dataframe
df_valid["semantic_similarity"] = similarities

# Display final table
df_valid[["Number", "semantic_similarity"]]
```

	Number	semantic_similarity
0	1.0	0.476901
1	2.0	0.558504

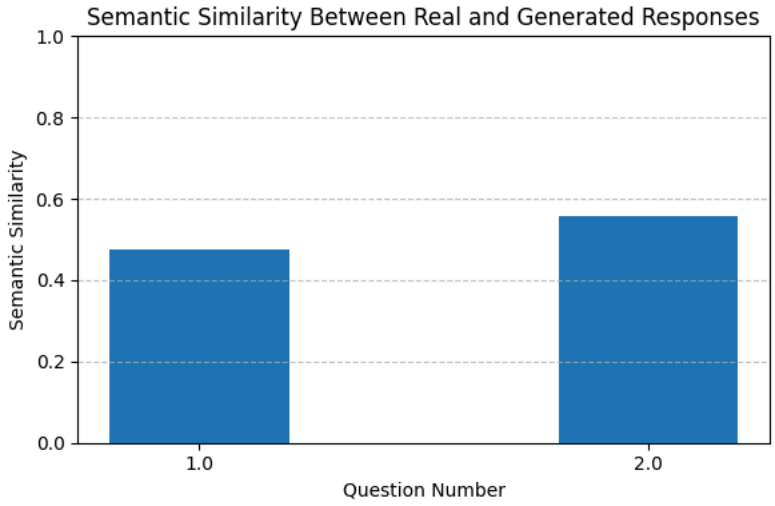
```
import matplotlib.pyplot as plt

# Plot semantic similarity by question
plt.figure(figsize=(6, 4))
plt.bar(df_valid["Number"], df_valid["semantic_similarity"], width=0.4)
plt.xlabel("Question Number")
plt.ylabel("Semantic Similarity")
plt.title("Semantic Similarity Between Real and Generated Responses")
plt.ylim(0, 1)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()

# Convert embedding tensors to lists
df_merged["embedding_real"] = [vec.tolist() for vec in emb_real]
df_merged["embedding_gen"] = [vec.tolist() for vec in emb_gen]

# Merge similarity scores from df_valid
df_export = pd.merge(df_merged, df_valid[["Number", "semantic_similarity"]], on="Number", how="left")

# Export to CSV
df_export.to_csv("/content/CDMX_Semantic_Embeddings.csv", index=False)
```



Haz doble clic (o pulsa Intro) para editar

```
from sentence_transformers import SentenceTransformer
import pandas as pd

# Load the model (si no lo has cargado antes)
model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")

# Asegúrate de tener bien cargado el CSV con los valores correctos
df = pd.read_csv("/content/CDMX_Sentiment_Comparation.csv", encoding="latin1")

# Clean and prepare
df["respuesta"] = df["respuesta"].astype(str)
df["Number"] = df["Number"].astype(str)

# Generate embeddings for each response individually
embeddings = model.encode(df["respuesta"].tolist(), convert_to_tensor=False)

# Add them as a new column
df["embedding"] = [e.tolist() for e in embeddings]

# Save the full DataFrame with embeddings
df.to_csv("/content/CDMX_AllResponses_withEmbeddings.csv", index=False)
```

```
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd

# Load data
df = pd.read_csv('/content/CDMX_Sentiment_Comparation.csv', encoding='latin1')
df = df.dropna(subset=['respuesta', 'Number', 'tipo'])
df["respuesta"] = df["respuesta"].astype(str)
df["Number"] = df["Number"].astype(str)

# Load sentence embedding model
model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")

results = []

# Loop over each question
for q in sorted(df['Number'].unique()):
    df_q = df[df['Number'] == q]

    # Get all real responses
    real_responses = df_q[df_q['tipo'] == 'real']['respuesta'].dropna().tolist()

    if len(real_responses) < 2:
        continue  # not enough data to compute reference

    # Create reference embedding for the full set of real responses
    real_reference_text = " ".join(real_responses)
    ref_embedding = model.encode(real_reference_text, convert_to_tensor=True).unsqueeze(0)
```

```
    # Compare each generated response to real reference
    for r in df_q[df_q['tipo'] == 'generada']['respuesta'].dropna().tolist():
        emb_r = model.encode(r, convert_to_tensor=True).unsqueeze(0)
        score = cosine_similarity(ref_embedding.cpu().numpy(), emb_r.cpu().numpy())[0][0]
        results.append({'tipo': 'generada', 'pregunta': int(float(q)), 'cosine_semantico': score})

    # Leave-one-out for real responses
    real_list = df_q[df_q['tipo'] == 'real']['respuesta'].dropna().tolist()
    for i, r in enumerate(real_list):
        other_real = real_list[:i] + real_list[i+1:]
        if not other_real:
            continue
        ref_text = " ".join(other_real)
        ref_emb = model.encode(ref_text, convert_to_tensor=True).unsqueeze(0)
        emb_r = model.encode(r, convert_to_tensor=True).unsqueeze(0)
        score = cosine_similarity(ref_emb.cpu().numpy(), emb_r.cpu().numpy())[0][0]
        results.append({'tipo': 'real', 'pregunta': int(float(q)), 'cosine_semantico': score})

# Save and display
df_sem_sim = pd.DataFrame(results)
df_sem_sim.to_csv('/content/semantic_similarity_vs_question_set.csv', index=False)
df_sem_sim.head()
```

	tipo	pregunta	cosine_semantico
0	generada	1	0.452662
1	generada	1	0.444602
2	generada	1	0.281238
3	generada	1	0.489437
4	generada	1	0.257603

```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import mannwhitneyu

# Get unique question numbers
questions = df_sem_sim['pregunta'].unique()

# Set up figure for subplots
fig, axes = plt.subplots(1, len(questions), figsize=(6 * len(questions), 6), sharey=True)

# Store results
stat_results = []

# Loop over each question
for i, q in enumerate(sorted(questions)):
    ax = axes[i] if len(questions) > 1 else axes

    # Filter data for this question
    df_q = df_sem_sim[df_sem_sim['pregunta'] == q]
    real_scores = df_q[df_q['tipo'] == 'real']['cosine_semantico']
    gen_scores = df_q[df_q['tipo'] == 'generada']['cosine_semantico']

    # Mann-Whitney U test
    u_stat, p_value = mannwhitneyu(real_scores, gen_scores, alternative='two-sided')
    stat_results.append({'pregunta': q, 'U': u_stat, 'p_value': p_value})

    # Boxplot + stripplot
    sns.boxplot(data=df_q, x='tipo', y='cosine_semantico', palette='pastel', ax=ax)
    sns.stripplot(data=df_q, x='tipo', y='cosine_semantico', color='black', alpha=0.4, ax=ax)

    ax.set_title(f'Pregunta {q}\nMann-Whitney p = {p_value:.4f}')
    ax.set_xlabel('Tipo de respuesta')
    ax.set_ylabel('Cosine Semántico' if i == 0 else '')
    ax.grid(axis='y', linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()

# Print test results
print("\nResultados estadísticos por pregunta:")
```

```
for r in stat_results:
    signif = "✅ SIGNIFICATIVA" if r['p_value'] < 0.05 else "🔍 No significativa"
    print(f"Pregunta {r['pregunta']}: U = {r['U']}, p = {r['p_value']:.4f} → {signif}")
```

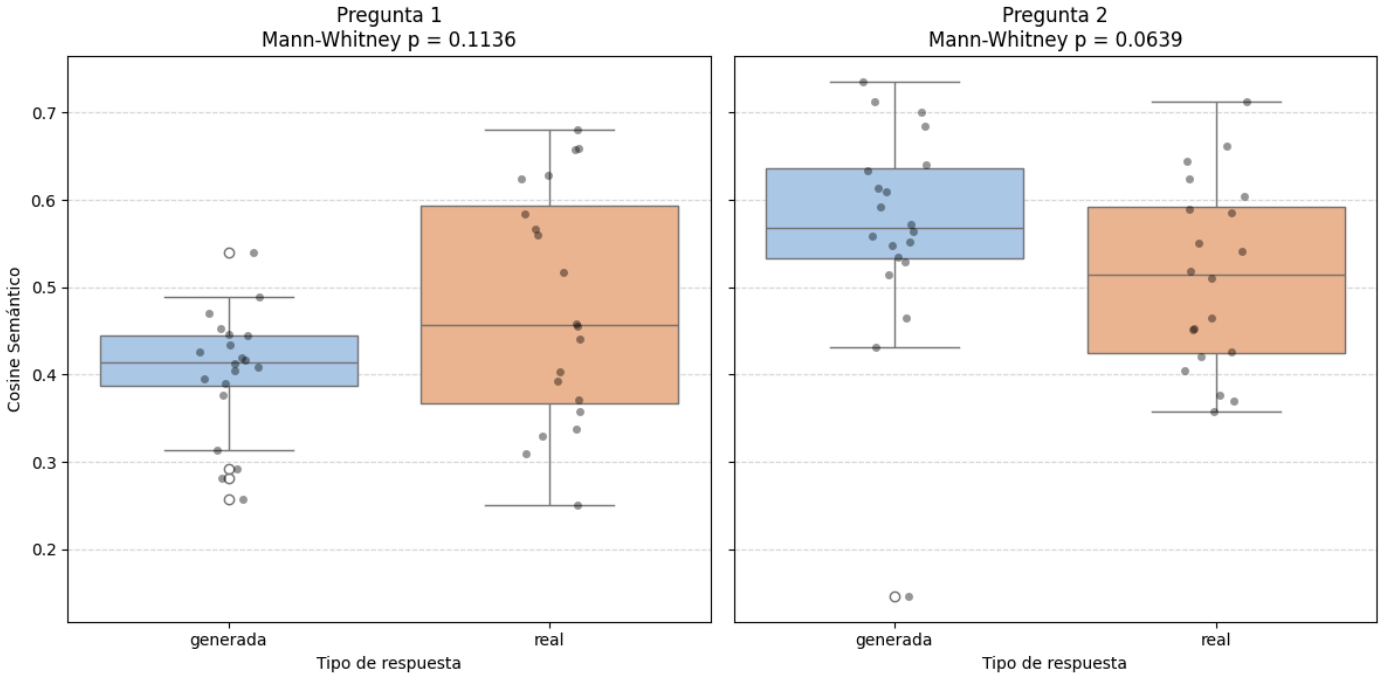
```
/tmp/ipython-input-8-4270462877.py:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.boxplot(data=df_q, x='tipo', y='cosine_semantico', palette='pastel', ax=ax)
/tmp/ipython-input-8-4270462877.py:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.boxplot(data=df_q, x='tipo', y='cosine_semantico', palette='pastel', ax=ax)
```




```
Resultados estadísticos por pregunta:
Pregunta 1: U = 259.0, p = 0.1136 → 🔍 No significativa
Pregunta 2: U = 131.0, p = 0.0639 → 🔍 No significativa
```