```python
# BLOQUE 0: Autenticación y conexión a Google Sheets

from google.colab import auth
from google.auth import default
from googleapiclient.discovery import build

# Autenticarse con tu cuenta de Google
auth.authenticate_user()
creds, _ = default()

# Crear servicio de acceso a Google Sheets
sheet = build('sheets','v4', credentials=creds).spreadsheets()

# ID de tu hoja de cálculo
SPREADSHEET_ID = '1O67tA_8QWbVzUa4TYylrpE1ER2S5GyEBB8svgKcO8C0' # Colocar aquí el ID de Google
```

```python
# BLOQUE 1: Obtener texto (desde enlace o texto manual)

import requests
from bs4 import BeautifulSoup

def extraer_texto_de_noticia(url):
    try:
        response = requests.get(url, timeout=10)
        response.raise_for_status()

        soup = BeautifulSoup(response.text, 'html.parser')
        parrafos = soup.find_all('p')
        texto = ' '.join([p.get_text() for p in parrafos])
        texto_limpio = ' '.join(texto.split())
        return texto_limpio

    except Exception as e:
        print("Error al procesar la noticia:", e)
        return None


# ===== ELEGIR UNA OPCIÓN =====

#  OPCIÓN A: usar enlace (dejá texto_manual = None)/OPCIÓN B: usar texto (dejá url = None)
url = None  #'https://thediplomat.com/2014/08/chinas-dangerous-intercept-of-us-spy-plane/'
texto_manual ="""

Chinese navy ship confronts U.S. cruiser amid tensions
David Lerman BLOOMBERG NEWS
Dec. 13, 2013Updated Dec. 14, 2013, 1:36 p.m. ET

A U.S. Navy guided-missile cruiser had a confrontation with a Chinese military ship on Dec. 5

The USS Cowpens, operating in international waters, and a Chinese naval vessel "had an encount

"This incident underscores the need to ensure the highest standards of professional seamanship

China was probably angry that the Cowpens may have been trying to spy on China's only aircraft

"This was not an accident," Cheng said in an interview. "It was deliberate. The Chinese are ra

The U.S. government lodged protests over the incident with Chinese officials in Beijing and Wa

The Chinese embassy in Washington didn't immediately respond to an e-mailed request for commer

The near-collision, while it was resolved peacefully, hints at the growing risk of confrontati

China last month unnerved its neighbors by declaring an air defense identification zone in the
```

```
49
50 'More Tension'
51
52 "I think we're going to see much more tension in the air and on the surface," Cheng said. "In
53
54 The Chinese vessel tried to force the Cowpens to stop, causing a military standoff, according
55
56 After a second Chinese ship sailed in front of the Cowpens and stopped, the U.S. vessel was fo
57
58 "It's getting dangerous out there," said Patrick Cronin, a senior adviser for the Asia-Pacific
59
60 Cronin said the incident appeared to be a "tit-for-tat" response to the U.S. refusing to recog
61
62 "We're making them look impotent with respect to the ADIZ," Cronin said, referring to the acro
63
64 By trying to block a U.S. ship, China is engaging in "coercive diplomacy – it's neither war no
65
66
67 """
68
69 # OPCIÓN B: pegar texto manual (dejá url = None)
70 # url = None
71 # texto_manual =
72
73
74 # ===== LÓGICA =====
75
76 if texto_manual and texto_manual.strip():
77     noticia = ' '.join(texto_manual.split())
78 elif url and url.strip():
79     noticia = extraer_texto_de_noticia(url)
80 else:
81     noticia = None
82
83 # Ver resumen sin que explote si falla
84 if noticia:
85     print(noticia[:500])
86 else:
87     print("⚠️ No se pudo obtener texto. Usá texto_manual o probá otro enlace.")
88
```

```
Chinese navy ship confronts U.S. cruiser amid tensions David Lerman BLOOMBERG NEWS Dec. 13, 2013Up
```

```
1 # BLOQUE2: Comparación de EFECTOS con el texto de la noticia (POR VENTANAS DE 2 ORACIONES, to
2
3 !pip install -q sentence-transformers
4
5 from sentence_transformers import SentenceTransformer, util
6 import torch
7 import re
8
9 modelo = SentenceTransformer('all-MiniLM-L6-v2')
10
11 # 1) Normalizar el texto (limpieza simple)
12 texto = ' '.join(noticia.split())
13
14 # 2) Separar en oraciones (corte simple por puntuación)
15 oraciones = [o.strip() for o in re.split(r'(?<=[\.\?\!])\s+', texto) if o.strip()]
16
17 # 3) Crear "ventanas" de 2 oraciones (chunking)
18 ventanas = []
19 for i in range(len(oraciones)):
20     ventana = " ".join(oraciones[i:i+2])  # 2 oraciones
21     if ventana:
22         ventanas.append(ventana)
23
```

```
# Si por alguna razón quedan muy pocas, usar el texto completo como respaldo
if len(ventanas) < 2:
    ventanas = [texto]

# Embeddings de cada ventana
embeddings_ventanas = modelo.encode(ventanas, convert_to_tensor=True)

# 4) Leer efectos estratégicos
result = sheet.values().get(
    spreadsheetId=SPREADSHEET_ID,
    range='List of Strategic Effects_cop!A2:H'
).execute()

values = result.get('values', [])

# 5) Calcular similitud (máxima) para cada efecto
correlaciones_efectos = []
MIN_SCORE_EFECTOS = 0.35
TOP_K_EFECTOS = 3

for fila in values:
    if len(fila) >= 8:
        efecto_texto = fila[3]
        embedding_efecto = modelo.encode(efecto_texto, convert_to_tensor=True)
        sims = util.pytorch_cos_sim(embeddings_ventanas, embedding_efecto).squeeze()
        similitud = float(torch.max(sims).item())

        # Guardar TODO (sin umbral fijo)
        correlaciones_efectos.append({
            "ID": fila[0],
            "Component": fila[1],
            "Intensity of Force": fila[2],
            "Strategic Effect": fila[3],
            "Col_E": fila[4],
            "Col_F": fila[5],
            "Col_G": fila[6],
            "Col_H": fila[7],
            "Similarity Score": round(similitud, 3)
        })

# Filtrar por mínimo + ordenar + tomar TOP 3
correlaciones_efectos = [c for c in correlaciones_efectos if c["Similarity Score"] >= MIN_SC(
correlaciones_efectos.sort(key=lambda x: x["Similarity Score"], reverse=True)
correlaciones_efectos = correlaciones_efectos[:TOP_K_EFECTOS]

# Resultado
# Resultado
if correlaciones_efectos:

    # Ordenar por Score de similitud (de mayor a menor)
    correlaciones_efectos.sort(
        key=lambda x: x.get("Similarity Score", 0),
        reverse=True
    )

    print(f"✅ Cantidad de correlaciones: {len(correlaciones_efectos)}")
    print("📌 Listado de efectos (ordenados por similitud):\n")

    for c in correlaciones_efectos:
        print(
            f"ID {c['ID']} – "
            f"{c['Strategic Effect']} "
            f"[{c['Intensity of Force']}] "
            f"(Score {c['Similarity Score']})"
        )

else:
```

```
    print("❌ No se encontraron correlaciones semánticas suficientes (≥ 0.35).")


```

```
WARNING:torchao.kernel.intmm:Warning: Detected no triton, on systems without Triton certain kernel
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: 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://huggingfac
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 datas
  warnings.warn(
modules.json: 100%  349/349 [00:00<00:00, 22.9kB/s]
config_sentence_transformers.json: 100%  116/116 [00:00<00:00, 8.66kB/s]
README.md: 10.5k/? [00:00<00:00, 770kB/s]
sentence_bert_config.json: 100%  53.0/53.0 [00:00<00:00, 5.46kB/s]
config.json: 100%  612/612 [00:00<00:00, 63.0kB/s]
model.safetensors: 100%  90.9M/90.9M [00:01<00:00, 102MB/s]
tokenizer_config.json: 100%  350/350 [00:00<00:00, 26.3kB/s]
vocab.txt: 232k/? [00:00<00:00, 6.11MB/s]
tokenizer.json: 466k/? [00:00<00:00, 15.5MB/s]
[illegible]00<00:00, 5.49kB/s]
config.json: 100%  190/190 [00:00<00:00, 8.46kB/s]
WARNING:google_auth_httplib2:httplib2 transport does not support per-request timeout. Set the time
WARNING:google_auth_httplib2:httplib2 transport does not support per-request timeout. Set the time
✅ Cantidad de correlaciones: 3
📌 Listado de efectos (ordenados por similitud):

ID 28 – Neutralised diplomatic cooperation [-0.6;-0.4] (Score 0.495)
ID 58 – Militarised disputed territory [-0.4;-0.2] (Score 0.494)
ID 83 – Information disseminated with a focus on disputed strategic territories [-0.2;0.0] (Score
```

```
# BLOQUE 3: Comparación con los indicadores (TOP 2 POR EFECTO EN 1 CELDA)

# Lista de IDs correlacionados desde la etapa anterior (NORMALIZADOS)
ids_detectados_set = set(str(correlacion["ID"]).strip() for correlacion in correlaciones_efec

# Leer hoja "Indicators"
result_ind = sheet.values().get(
    spreadsheetId=SPREADSHEET_ID,
    range='Indicators!A2:C'
).execute()

valores_indicadores = result_ind.get('values', [])

# Guardamos indicadores por ID de efecto
indicadores_por_id = {}  # clave: ID efecto, valor: lista de indicadores (con score)

for fila in valores_indicadores:
    if len(fila) >= 3:
        id_indicador = str(fila[0]).strip()
        tipo_indicador = fila[1]
        texto_indicador = fila[2]

        # ✅ FILTRO ESTRICTO: solo evaluar indicadores cuyo ID esté en los efectos detectado
        if id_indicador not in ids_detectados_set:
            continue

        embedding_indicador = modelo.encode(texto_indicador, convert_to_tensor=True)
        sims = util.pytorch_cos_sim(embeddings_ventanas, embedding_indicador).squeeze()
```

```
        similitud = float(torch.max(sims).item())

        if similitud >= 0.2:  # Umbral ajustable
            if id_indicador not in indicadores_por_id:
                indicadores_por_id[id_indicador] = []

            indicadores_por_id[id_indicador].append({
                "Type": tipo_indicador,
                "Text": texto_indicador,
                "Score": round(similitud, 3)
            })

# Armar 1 texto por ID con TOP 2 indicadores
indicadores_texto_por_id = {}  # clave: ID efecto, valor: string con TOP2
for id_efecto, lista in indicadores_por_id.items():
    lista_ordenada = sorted(lista, key=lambda x: x["Score"], reverse=True)[:2]
    texto_celda = "\n".join([f"{x['Type']}: {x['Text']} (Score {x['Score']})" for x in lista_
    indicadores_texto_por_id[id_efecto] = texto_celda

# (Opcional) Mostrar por pantalla
if indicadores_texto_por_id:
    # IDs que sí tienen indicadores (para condición de suficiencia en el bloque final)
    ids_con_indicadores = set(indicadores_texto_por_id.keys())

    print("✅ TOP 2 indicadores por efecto (en 1 celda):")
    for k, v in indicadores_texto_por_id.items():
        print(f"\nID {k}:\n{v}")
else:
    ids_con_indicadores = set()
    print("❌ No se encontraron indicadores correlacionados. El proceso se detiene.")

```

```
✅ TOP 2 indicadores por efecto (en 1 celda):

ID 28:
Diplomatic: Increase in domestic political unrest or protests due to perceived diplomatic failure
Military: Use of military diplomacy to rebuild or strengthen international relationships post-neut

ID 58:
Military: Strategic use of air, naval, and ground forces to assert territorial claims in disputed
Military: Use of military diplomacy to assert territorial claims and de-escalate tensions in milit

ID 83:
Military: Military involvement in securing borders and access points related to disputed strategic
Diplomatic: Creation of narratives highlighting historical ties and sovereign claims over disputed
```

```
# BLOQUE 4: Comparación con los medios y modos (Means y Ways independientes)
# Regla: HASTA 4 resultados por lista, SOLO si superan el umbral mínimo

# Parámetros del bloque
MIN_SCORE_MM = 0.35   # umbral mínimo de relevancia
TOP_K_MM = 4          # HASTA 4 (no obligatorio llegar a 4)

medios_detectados = []
modos_detectados = []

def detectar_y_guardar_medio_modo(entrada, tipo, es_medio=True):
    entrada_txt = str(entrada).strip()
    if not entrada_txt:
        return

    embedding_item = modelo.encode(entrada_txt, convert_to_tensor=True)
    sims = util.pytorch_cos_sim(embeddings_ventanas, embedding_item).squeeze()
    similitud = float(torch.max(sims).item())

    item = {
        "Tipo": tipo,
```

```
        "Texto": entrada_txt,
        "Score": round(similitud, 3)
    }

    if es_medio:
        medios_detectados.append(item)
    else:
        modos_detectados.append(item)

# --- Diplomatic ---
political_data = sheet.values().get(
    spreadsheetId=SPREADSHEET_ID,
    range='Diplomatic Ways & Means!A2:B'
).execute().get('values', [])

for fila in political_data:
    if len(fila) >= 1 and fila[0]:
        detectar_y_guardar_medio_modo(fila[0], "Diplomatic", es_medio=True)
    if len(fila) >= 2 and fila[1]:
        detectar_y_guardar_medio_modo(fila[1], "Diplomatic", es_medio=False)

# --- Economic ---
economic_data = sheet.values().get(
    spreadsheetId=SPREADSHEET_ID,
    range='Economic Ways & Means!A2:B'
).execute().get('values', [])

for fila in economic_data:
    if len(fila) >= 1 and fila[0]:
        detectar_y_guardar_medio_modo(fila[0], "Economic", es_medio=True)
    if len(fila) >= 2 and fila[1]:
        detectar_y_guardar_medio_modo(fila[1], "Economic", es_medio=False)

# --- Military ---
military_data = sheet.values().get(
    spreadsheetId=SPREADSHEET_ID,
    range='Military Ways & Means!A2:B'
).execute().get('values', [])

for fila in military_data:
    if len(fila) >= 1 and fila[0]:
        detectar_y_guardar_medio_modo(fila[0], "Military", es_medio=True)
    if len(fila) >= 2 and fila[1]:
        detectar_y_guardar_medio_modo(fila[1], "Military", es_medio=False)

# --- Informational ---
informational_data = sheet.values().get(
    spreadsheetId=SPREADSHEET_ID,
    range='Informational Ways & Means!A2:B'
).execute().get('values', [])

for fila in informational_data:
    if len(fila) >= 1 and fila[0]:
        detectar_y_guardar_medio_modo(fila[0], "Informational", es_medio=True)
    if len(fila) >= 2 and fila[1]:
        detectar_y_guardar_medio_modo(fila[1], "Informational", es_medio=False)

# --- Selección final: umbral + HASTA TOP_K ---
medios_detectados = [m for m in medios_detectados if m["Score"] >= MIN_SCORE_MM]
modos_detectados  = [w for w in modos_detectados  if w["Score"] >= MIN_SCORE_MM]

medios_detectados.sort(key=lambda x: x["Score"], reverse=True)
modos_detectados.sort(key=lambda x: x["Score"], reverse=True)

medios_detectados = medios_detectados[:TOP_K_MM]
modos_detectados  = modos_detectados[:TOP_K_MM]

```

```
# --- Resultados ---
print("\n✅ Medios (Means) detectados (HASTA 4, ordenados por similitud):")
for m in medios_detectados:
    print(f"→ ({m['Tipo']}) {m['Texto']} [Score: {m['Score']}]")

print("\n✅ Modos (Ways) detectados (HASTA 4, ordenados por similitud):")
for w in modos_detectados:
    print(f"→ ({w['Tipo']}) {w['Texto']} [Score: {w['Score']}]")

```

```
✅ Medios (Means) detectados (HASTA 4, ordenados por similitud):
→ (Military) Warships [Score: 0.49]
→ (Military) Navy [Score: 0.475]
→ (Military) Target maritime spaces [Score: 0.474]
→ (Military) Security forces [Score: 0.41]

✅ Modos (Ways) detectados (HASTA 4, ordenados por similitud):
→ (Military) Crisis response operations [Score: 0.513]
→ (Military) Counterinsurgency operations [Score: 0.511]
→ (Informational) Force confrontation [Score: 0.469]
→ (Military) Assassination operations [Score: 0.45]
```

```
import re
from datetime import datetime

# --- Datos del hecho ---
fact_id = "10"  # Esto luego lo podemos automatizar
fecha_analisis = datetime.today().strftime('%Y-%m-%d')
resumen = noticia[:250] + "."

# --- URL del hecho (carga manual si la noticia se pegó a mano) ---
url_manual = "https://www.telegram.com/story/news/state/2013/12/14/chinese-navy-ship-confror

# URL efectiva para exportación (siempre queda algo)
if url_manual:
    url_export = url_manual
else:
    url_export = url if 'url' in globals() and url else "No URL"

# --- Fecha del hecho (carga manual) ---
# 👉 completar siempre antes de ejecutar

mes_hecho  = "12"    # ejemplo: "06", sin fecha:"n.d."
anio_hecho = "2012"    # ejemplo: "2023"

if mes_hecho and anio_hecho:
    fecha_hecho = f"{anio_hecho}-{mes_hecho}"
else:
    fecha_hecho = "No date registered"

# --- Agrupar medios / modos por componente en una sola celda (con saltos de línea) ---
def agrupar_por_tipo(lista_items):
    # Orden de encabezados por componente (DIME)
    orden_tipos = ["Military", "Diplomatic", "Economic", "Informational"]

    grupos = {}
    for it in lista_items:
        grupos.setdefault(it["Tipo"], []).append(it)

    lineas = []
    for tipo in orden_tipos:
        if tipo in grupos and grupos[tipo]:
            lineas.append(tipo)
            # dentro del tipo, ordenar por Score desc
            for it in sorted(grupos[tipo], key=lambda x: x.get("Score", 0), reverse=True):
                lineas.append(f"{it['Texto']} (Score {it['Score']})")
```

```
    return "\n".join(lineas)


# --- Construcción de filas resultado ---
filas_resultado = []

# ✅ condición de suficiencia global: si no hay medios o no hay modos, no exportar nada
if not medios_detectados or not modos_detectados:
    print("❌ No se exporta: faltan Means o Ways.")
else:
    means_celda = agrupar_por_tipo(medios_detectados)
    ways_celda  = agrupar_por_tipo(modos_detectados)

    for efecto in correlaciones_efectos:
        efecto_id = efecto["ID"]
        efecto_component = efecto["Component"]

        # ✅ condición de suficiencia por efecto:
        # si este efecto NO tiene indicadores, NO sigue y NO exporta
        if efecto_id not in ids_con_indicadores:
            continue

        efecto_nombre = f"{efecto['Strategic Effect']} (Score {efecto['Similarity Score']})"
        intensidad = efecto["Intensity of Force"]

        # Indicadores TOP 2 (1 celda) para este efecto
        indicadores_celda = indicadores_texto_por_id[efecto_id]

        # ✅ 1 fila por efecto (ya NO se combinan medios x modos)
        fila = [
            fact_id,
            url_export,
            fecha_analisis,
            fecha_hecho,
            resumen,
            efecto_id,
            intensidad,
            efecto_component,
            efecto_nombre,
            indicadores_celda,
            ways_celda,
            means_celda,
            efecto["Col_E"],
            efecto["Col_F"],
            efecto["Col_G"],
            efecto["Col_H"]
        ]
        filas_resultado.append(fila)

# --- Exportar a la hoja "Code_Results" en modo acumulativo ---
if filas_resultado:
    sheet.values().append(
        spreadsheetId=SPREADSHEET_ID,
        range='Code_Output!A1',
        valueInputOption='RAW',
        insertDataOption='INSERT_ROWS',
        body={'values': filas_resultado}
    ).execute()
    print(f"✅ Exportadas {len(filas_resultado)} filas con enlace de fuente incluido.")
else:
    print("⚠️ No se exportó nada: no hubo efectos que cumplieran suficiencia (Indicadores +


```

```
✅ Exportadas 3 filas con enlace de fuente incluido.
```