Proyecto Final: "Análisis NLP para la Cohesión en Equipos de Trabajo"

Presenta: Ignacio David Vázquez Pérez
Matrícula: 20182928
Programa: Ingeniería en Computación
Asesora: ADRIANA PENA PEREZ NEGRON
Titulares del Diplomado:
M. en C. Tania Gisela Alcántara Medina
M. en C. Fernando Javier Aguilar Canto
M. en C. José Alberto Torres León
Fecha de Entrega: 22 de octubre de 2025

Índice

7. **Referencias Bibliográficas**
8. **Anexos**
    - 8.1. Datos Resumidos (Muestra)

1. Introducción

1.1. Contexto del Problema

La "cohesión de equipo" es un factor crítico para el éxito organizacional, pero es intrínsecamente difícil de medir. Tradicionalmente, las empresas dependen de encuestas de clima laboral con escalas numéricas (ej. 1 al 5) o entrevistas cualitativas. Las respuestas numéricas carecen de matices, mientras que el análisis de respuestas abiertas (texto libre) es un proceso manual, lento y subjetivo, que consume cientos de horas de recursos humanos [cite: reportes_bimnestrales.docx].

1.2. Importancia del Proyecto

Este proyecto aborda la necesidad de un análisis objetivo, escalable y profundo de la cohesión de equipo. Su importancia radica en:

- **Eficiencia:** Automatiza el análisis de grandes volúmenes de texto (respuestas de encuestas, comentarios) mediante un pipeline de Procesamiento de Lenguaje Natural (PLN).
- **Profundidad de Análisis:** Va más allá de las palabras clave, utilizando análisis de sentimiento para capturar el tono emocional y modelos avanzados de Transformers para resumir y extraer los temas centrales de las respuestas.
- **Inteligencia Accionable:** Al agrupar los resultados por "Años de Experiencia" o "Empresas", el sistema permite a la organización identificar patrones específicos y tendencias en diferentes cohortes, generando insights que serían imposibles de obtener manualmente.

1.3. Objetivos Específicos (Actualizado)

1. **Objetivo 1:** Recolectar y estructurar las respuestas cualitativas de los profesionales (de respuestas_d.docx y audios_escritos.csv) y estructurarlas en un conjunto de datos unificado (output.csv).
2. **Objetivo 2:** Desarrollar un pipeline de PLN exploratorio (NLPAnalysis) para realizar un análisis inicial de frecuencia (n-gramas), sentimiento y lematización sobre el texto (basado en nlp_servicio_social.ipynb y nlp_servicio_social.pdf).
3. **Objetivo 3:** Implementar un modelo de **Resumen de Texto** (Summarization) basado en Transformers para destilar las respuestas largas en temas clave y accionables (basado en summarized_answers.csv y factors_summary_5.txt).
4. **Objetivo 4:** Agrupar y presentar los resultados tanto del análisis exploratorio como del resumen por categorías demográficas (Años de Experiencia, Empresa).

1.4. Alcance y Limitaciones

- **Alcance:** El sistema procesa un corpus de texto en español (output.csv). Primero, aplica un análisis estadístico (NLPAnalysis) para gráficas e histogramas. Segundo, aplica un modelo de resumen de texto para extraer los temas principales de todas las respuestas (summarized_answers.csv).
- **Limitaciones:** El éxito de las técnicas de PLN depende de la calidad y cantidad del texto. Como se demostró en la Fase 1, el intento de usar **Word Embedding falló** debido a que las respuestas individuales eran demasiado cortas (Not enough samples or features) [cite: nlp_servicio_social.ipynb, nlp_servicio_social.pdf]. Esto validó la necesidad de usar un enfoque de Resumen (Summarization) en la Fase 2, que resultó ser mucho más efectivo.

2. Marco Teórico

2.1. Procesamiento de Lenguaje Natural (PLN)

El PLN es una rama de la IA que permite a las máquinas leer, comprender e interpretar el lenguaje humano. En este proyecto, se utiliza para transformar cientos de respuestas de texto no estructurado en insights estructurados.

2.2. Lematización y Análisis de Frecuencia (N-gramas)

- **Lematización:** Es una técnica de normalización de texto que reduce las palabras a su forma base (lema) usando un diccionario (ej. "comunicación", "comunicando" -> "comunicar"). Es más preciso que la estematización (stemming) y es fundamental para agrupar palabras conceptualmente [cite: nlp_servicio_social.ipynb].
- **Análisis de Frecuencia (N-gramas):** Se utiliza para generar histogramas de las palabras más comunes (1-grama) y combinaciones de palabras (2-gramas, 3-gramas). Esto proporciona una visión inmediata de los temas dominantes, como "buena comunicación" [cite: nlp_servicio_social.ipynb, nlp_servicio_social.pdf].

2.3. Análisis de Sentimiento

Es una técnica de PLN usada para determinar el tono emocional de un texto (positivo, negativo o neutral) [cite: nlp_servicio_social.ipynb]. Es crucial para este proyecto, ya que permite cuantificar la "cohesión" no solo por *qué* dicen los empleados, sino por *cómo* lo dicen, detectando frustración o satisfacción.

2.4. Resumen de Texto (Summarization) con Transformers

Esta es la técnica central de la Fase 2 del proyecto y la que aporta mayor valor.

- **El Problema:** El análisis de frecuencia (Fase 1) solo nos dice las palabras más comunes (ej. "comunicación"), pero no el *contexto* o la *idea* detrás de ellas. El Word Embedding falló por datos escasos [cite: nlp_servicio_social.ipynb, nlp_servicio_social.pdf].

- **La Solución (Summarization):** Se utiliza un modelo de lenguaje avanzado de la familia Transformers (como BERT, BART o T5) para leer *todas* las respuestas a una pregunta y generar un resumen que condensa las ideas principales.
- **Resumen Abstractivo (Usado Aquí):** El modelo "entiende" el significado de todas las respuestas y *genera nuevo texto* que describe los temas centrales. Esto es lo que vemos en archivos como factors_summary_5.txt y summarized_answers.csv.
- **Detalle de Implementación:** El proceso implica tomar todas las respuestas a una pregunta (ej. todos los 61 "Factores" de respuestas_d.docx), concatenarlas en un solo documento grande. Para manejar las limitaciones de memoria de la GPU, este documento se divide en "batches" (lotes). Cada lote se resume de forma independiente, y luego los resúmenes intermedios se concatenan y se resumen una vez más para obtener el resumen final.

3. Metodología

3.1. Descripción General del Proceso (En dos fases)

El proyecto se dividió en dos fases metodológicas:

1. **Fase 1: Análisis Exploratorio (NLPAnalysis):** Se desarrolló una clase de Python (NLPAnalysis) para realizar un análisis estadístico rápido del corpus [cite: nlp_servicio_social.ipynb]. Esto incluyó la lematización, la generación de histogramas de frecuencia de n-gramas, análisis de sentimiento y un intento de Word Embedding.
2. **Fase 2: Análisis de Resumen (Transformers):** Al descubrir que Word Embedding fallaba por falta de datos, la metodología cambió. Se adoptó un enfoque de resumen de texto (Summarization) para tomar la totalidad de las respuestas por pregunta, procesarlas en lotes (batches) para gestionar la memoria de la GPU, y destilarlas en sus temas principales usando un modelo pre-entrenado de Transformers.

3.2. Datos: Origen, Descripción y Preguntas de la Audiencia

Las fuentes de datos para este proyecto fueron recopiladas a través de encuestas y entrevistas a 61 profesionales [cite: reportes_bimnestrales.docx, audios_escritos.csv].

- **Preguntas de la Audiencia:** La encuesta incluyó las siguientes preguntas clave para evaluar la cohesión [cite: reportes_bimnestrales.docx, audios_escritos.csv]:
    1. ¿Cuántos años de experiencia tienes en el desarrollo de software?
    2. ¿En qué empresa trabajas?
    3. ¿Cuál es tu puesto?
    4. ¿Cuál es para ti el significado de cohesión en un equipo de trabajo?
    5. Describe los factores que consideres contribuyen al desarrollo de cohesión en un equipo.
    6. Cita alguna situación que creas refleje este tipo de cohesión.
    7. Hay algún incidente que recuerdes que refleja mala o poca cohesión.
- **Datos Crudos:** respuestas_d.docx [cite: respuestas_d.docx] y audios_escritos.csv [cite: audios_escritos.csv]. Contienen las transcripciones completas de las entrevistas.

- **Datos Estructurados:** output.csv [cite: output.csv]. Una versión limpia de las respuestas, utilizada para la Fase 1.
- **Datos Resumidos (Salida):** summarized_answers.csv [cite: summarized_answers.csv] y factors_summary_...txt [cite: factors_summary_4.txt, factors_summary_5.txt]. Estos archivos contienen los resúmenes generados por el modelo de Transformers.

3.3. Arquitectura del Modelo

La arquitectura es un pipeline híbrido:

1. **Módulo 1: NLPAnalysis (Script de Python)**
   - **Entrada:** output.csv (DataFrame).
   - **Proceso:** Carga spacy para lematización. Usa matplotlib y seaborn para crear histogramas y gráficas de n-gramas (1, 2 y 3 palabras).
   - **Salida:** Gráficas de frecuencia y el hallazgo del fallo de Word Embedding.
2. **Módulo 2: Summarization (Pipeline de Transformers)**
   - **Entrada:** Texto concatenado de una columna (ej. todas las 61 respuestas de "Factores" de respuestas_d.docx).
   - **Proceso:** Un modelo de resumen (ej. facebook/bart-large-cnn o un modelo en español) lee el texto completo. Para evitar errores de memoria de la GPU, el texto largo se divide en lotes (batches) de un tamaño máximo de tokens (ej. 1024 tokens), cada lote se resume, y los resúmenes resultantes se vuelven a concatenar y resumir.
   - **Salida:** Un resumen de texto coherente que identifica los temas principales.

3.4. Herramientas y Tecnologías

- **Núcleo de Análisis:** pandas, spacy (para lematización).
- **Análisis Avanzado:** transformers (de Hugging Face) para el pipeline de Summarization.
- **Visualización:** matplotlib, seaborn.
- **Entorno:** Jupyter Notebook (.ipynb).

4. Implementación (Análisis Técnico Detallado)

La implementación del proyecto se dividió en dos fases técnicas claras.

4.1. Fase 1: Análisis Exploratorio con NLPAnalysis

Esta fase, documentada en el nlp_servicio_social.ipynb y nlp_servicio_social.pdf, consistió en crear una clase de Python para realizar un análisis estadístico inicial y explorar la viabilidad de técnicas de PLN clásicas. El código central es el siguiente:

```
# Importaciones clave de la Fase 1
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
from nltk.stem import SnowballStemmer, WordNetLemmatizer
import string
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from textblob import TextBlob
from gensim.models import Word2Vec
import spacy

class NLPAnalysis:
    """
    Clase para realizar un análisis NLP exploratorio de las respuestas de la encuesta.
    Incluye preprocesamiento, análisis de sentimiento, frecuencia y word embedding.
    """

    def __init__(self, df, column_names, remove_sentiment_stop_words=False):
        """
        Inicializa la clase con el DataFrame y las columnas a analizar.
        """
        self.df = df
        self.column_names = column_names
        # Carga el modelo español de spaCy para lematización
        self.nlp = spacy.load("es_core_news_sm")
        self.lemmatizer = WordNetLemmatizer()
        self.stemmer = SnowballStemmer('spanish')
        self.stop_words = set(stopwords.words('spanish'))
        self.remove_sentiment_stop_words = remove_sentiment_stop_words

    def preprocess_text(self, text, remove_stop_words=True):
        """
        Limpia y lematiza el texto.
        1. Tokeniza, 2. Elimina puntuación, 3. Lematiza (usando spaCy).
        """
        if not isinstance(text, str):
            text = str(text)

        doc = self.nlp(text.lower())
        lematized_tokens = []
        for token in doc:
            if not token.is_punct and not token.is_space:
                if remove_stop_words and token.lemma_ in self.stop_words:
                    continue
                lematized_tokens.append(token.lemma_)
```

```
        return lematized_tokens

    def get_sentiment(self, text):
        """
        Utiliza TextBlob para un análisis de sentimiento rápido.
        Nota: TextBlob tiene un soporte limitado para español.
        """
        analysis = TextBlob(text)
        return analysis.sentiment.polarity

    def plot_word_frequency(self, column_name, top_n=20, n_gram=1):
        """
        Calcula la frecuencia de N-gramas (1, 2, o 3 palabras) y
        genera un histograma con matplotlib/seaborn.
        """
        all_tokens = [token for text in self.df[column_name].dropna() for token in
self.preprocess_text(text)]

        if n_gram > 1:
            # Genera n-gramas (tuplas de palabras)
            ngram_counts = Counter(nltk.ngrams(all_tokens, n_gram))
            common_ngrams = ngram_counts.most_common(top_n)
            print(f"Combinaciones de {n_gram} palabras más importantes para {column_name}:
{common_ngrams}")
        else:
            # Conteo de palabras individuales
            word_freq = Counter(all_tokens)
            common_words = word_freq.most_common(top_n)
            print(f"Combinaciones de 1 palabra más importantes para {column_name}:
{common_words}")

        # ... (código para generar el gráfico de barras con plt.bar()) ...

    def word_embedding_analysis(self, column_name):
        """
        Intenta entrenar un modelo Word2Vec con el texto de la columna.
        Este es el método que falla si los datos son insuficientes.
        """
        tokenized_data = [self.preprocess_text(text) for text in self.df[column_name].dropna()]

        # Filtro para evitar que Word2Vec falle con oraciones vacías
        processed_data = [s for s in tokenized_data if s]
```

```
        if len(processed_data) < 5 or len(processed_data[0]) < 2:
            print(f"Not enough samples or features for word embedding in {column_name}")
            return

        # Entrena el modelo Word2Vec
        model = Word2Vec(processed_data, vector_size=100, window=5, min_count=1,
workers=4)
        print(f"Word Embedding - Palabras similares a 'comunicación' en {column_name}:")
        try:
            print(model.wv.most_similar('comunicación', topn=5))
        except KeyError:
            print("La palabra 'comunicación' no está en el vocabulario.")

    def run(self):
        """
        Ejecuta el pipeline de análisis completo para todas las columnas.
        """
        for col in self.column_names:
            print(f"--- Iniciando Análisis para: {col} ---")
            # 1. Análisis de Frecuencia
            self.plot_word_frequency(col, n_gram=1)
            self.plot_word_frequency(col, n_gram=2)
            self.plot_word_frequency(col, n_gram=3)

            # 2. Análisis de Sentimiento (ejemplo)
            # self.df[f'{col}_sentiment'] = self.df[col].apply(self.get_sentiment)

            # 3. Word Embedding
            self.word_embedding_analysis(col)
```

4.2. Fase 2: Resumen de Texto con Modelos Transformers

El fallo de word_embedding_analysis en la Fase 1 demostró que las respuestas individuales eran demasiado cortas para entrenar un modelo. La solución fue cambiar a un modelo **Transformers pre-entrenado** (como BART o T5) y usarlo para **resumen abstractivo**.

A diferencia de Word2Vec, que aprende desde cero, un modelo de Transformers ya ha sido entrenado en miles de millones de textos y "entiende" el lenguaje. Se utiliza para leer *todas* las respuestas de una pregunta como un solo documento grande y generar un resumen coherente.

4.2.1. El Desafío: Límites de Memoria de la GPU y Contexto

Los modelos Transformers son potentes pero tienen una limitación clave: una **ventana de contexto máxima** (ej. 1024 o 2048 tokens). Si concatenamos las 61 respuestas a la Pregunta 5 (Factores), el texto resultante es mucho más largo que este límite.

Intentar procesar un texto que excede la ventana de contexto resultaría en un error de CUDA out of memory (si se usa GPU) o simplemente en el truncamiento del texto, perdiendo información valiosa.

4.2.2. La Solución: Resumen en Lotes (Batches)

Para solucionar esto, se implementó un **proceso de resumen recursivo en lotes (batches)**, gestionando así el uso de memoria:

1. **Concatenación:** Se tomaron todas las 61 respuestas de una pregunta (ej. "Factores") y se unieron en un solo string de texto gigante (full_text).
2. **División en Lotes (Batching):** Se dividió full_text en una lista de lotes (batches), donde cada lote tenía un tamaño máximo de tokens (ej. 1000 tokens) para caber cómodamente en la memoria de la GPU.
3. **Resumen Intermedio:** Se iteró sobre la lista de lotes y se resumió cada uno por separado.
    - summary_batch_1 = model.summarize(batch_1)
    - summary_batch_2 = model.summarize(batch_2)
    - ...
4. **Concatenación Intermedia:** Los resúmenes resultantes (que son mucho más cortos que los lotes originales) se unieron en un nuevo string (intermediate_summary_text).
5. **Resumen Final:** Se ejecutó el modelo de resumen una última vez sobre el intermediate_summary_text para crear el resumen final y cohesivo que se guardó en summarized_answers.csv.

Este enfoque aseguró que todo el contexto del documento original fuera considerado, sin exceder los límites de hardware de la GPU.

5. Resultados y Discusión

5.1. Hallazgos de la Fase 1 (El Fallo de Word Embedding)

El hallazgo más importante de la Fase 1 fue un fallo controlado. Al intentar ejecutar el análisis de Word Embedding sobre la columna 'Factores', el sistema arrojó el siguiente error:

> Not enough samples or features for word embedding in Factores
> [cite: nlp_servicio_social.ipynb, nlp_servicio_social.pdf]

- **Interpretación:** Esto demuestra que las respuestas individuales, aunque numerosas (61), son demasiado cortas y carecen de la riqueza de vocabulario necesaria para entrenar un modelo de embedding desde cero.
- **Impacto en la Metodología:** Este fallo fue crucial, ya que invalidó el enfoque de análisis

de similitud a nivel de palabra y **validó la necesidad de cambiar a un enfoque de resumen (Summarization)**, que analiza el corpus en su totalidad en lugar de palabras individuales.

5.2. Resultados de la Fase 2 (Resúmenes de Temas Clave)

Esta fase fue exitosa. Al aplicar el modelo de resumen de Transformers a las respuestas de la audiencia, se obtuvieron los siguientes temas clave [cite: summarized_answers.csv, factors_summary_4.txt, factors_summary_5.txt]:

Pregunta 4: ¿Qué es la cohesión para ti en un equipo de trabajo?

Fuente: summarized_answers.csv

- **Integración y Ambiente:** Se define como una buena integración del equipo y un buen ambiente de trabajo.
- **Conocimiento Compartido:** Implica que todos los miembros del equipo saben cuál es su trabajo y también el trabajo de los demás.
- **Comunicación y Dirección:** Requiere una comunicación clara y constante, asegurando que todos tengan la misma dirección y compartan un flujo de trabajo.
- **Claridad de Roles:** Es fundamental tener roles bien definidos.
- **Metas Comunes:** Un equipo cohesionado es aquel que cumple sus objetivos y posee el conocimiento adecuado para lograrlos.

Pregunta 5: Factores que contribuyen a la cohesión

Fuente: factors_summary_5.txt, summarized_answers.csv

- **Comunicación (Dominante):** Es el factor más crítico. Se describe como "buena comunicación", "clara", "directa", "abierta" y "de confianza".
- **Ambiente y Relaciones:** Un "buen ambiente de trabajo", "compañerismo", "empatía", "confianza", "responsabilidad" y "compromiso" son esenciales.
- **Claridad y Liderazgo:** Se necesita una "buena organización", "liderazgo", "tareas bien detalladas" (claridad) y "roles bien definidos".
- **Convivencia:** Se mencionan las actividades *fuera* del trabajo (convivencia, tiempo libre) como un factor importante para generar empatía.
- **Apertura:** Tener una "mente abierta", "apertura para dar y pedir ayuda" y "saber cómo pedirla y darla".

Pregunta 6: Nombra una situación que CREA cohesión

Fuente: summarized_answers.csv

- **Resolución de Problemas:** Reunirse para detectar un fallo del sistema y buscar la solución colectivamente, en lugar de señalar culpables.
- **Colaboración:** Distribuir la carga de trabajo y usar el trabajo previo como referencia.
- **Defensa del Equipo:** Tener que resolver un asunto del equipo frente a otra área de trabajo.

- **Soporte en Errores:** Apoyarse mutuamente cuando ocurren errores y tener una buena estructura de requerimientos.

Pregunta 7: Nombra una situación que refleje MALA cohesión

Fuente: summarized_answers.csv

- **Mala Comunicación:** Falta de comunicación y organización en la priorización de tareas; "hacer un poco de todo" en lugar de seguir un orden.
- **Metas Irreales:** Cuando un equipo (ej. Ventas) impone metas no realistas a otro (ej. Desarrollo), generando estrés excesivo.
- **Falta de Colaboración:** Personas o personalidades que "rehúsan trabajar en equipo".
- **Suposiciones:** Asumir lo que el cliente o el usuario quiere en lugar de preguntar, lo que genera discusiones internas.
- **Entregas Deficientes:** "Mal delivery" (entregas de mala calidad).

5.3. Interpretación de los Resultados

El análisis de Fase 1 (frecuencia) nos dijo que "comunicación" era la palabra más importante (38 menciones) [cite: nlp_servicio_social.ipynb]. El análisis de Fase 2 (resumen de Transformers) nos dio el **contexto accionable**: la comunicación debe ser "clara", "directa" y "de confianza", y su ausencia es la principal causa de mala cohesión, llevando a "estrés" y "prioridades desorganizadas".

El resumen de Fase 2 fue capaz de capturar matices que el análisis de frecuencia no pudo, como la importancia de las "actividades fuera del trabajo" o el conflicto entre "metas no realistas" y "estrés". El resultado de la Fase 2 es, por lo tanto, significativamente más valioso para la organización.

6. Conclusiones

6.1. Resumen de Resultados

El proyecto logró transformar un gran volumen de datos cualitativos (respuestas_d.docx, audios_escritos.csv) en insights cuantitativos y cualitativos. La Fase 1 (NLPAnalysis) fue exitosa para análisis de frecuencia, pero demostró que el corpus de texto no era adecuado para entrenar un modelo de Word Embedding. La Fase 2 (Resumen con Transformers) superó esta limitación y proporcionó resúmenes detallados y coherentes de los temas clave, identificando la "comunicación clara" como el pilar de la cohesión y la "falta de organización" y "metas irreales" como sus principales detractores.

6.2. Cumplimiento de Objetivos

- **Objetivo 1 (Recolectar Datos):** Cumplido. (Archivos .docx y .csv).
- **Objetivo 2 (Pipeline Exploratorio):** Cumplido. (Clase NLPAnalysis y sus resultados).
- **Objetivo 3 (Resumen BERT):** Cumplido. (Archivos factors_summary_5.txt y

summarized_answers.csv).
- **Objetivo 4 (Agrupación):** Cumplido. (El NLPAnalysis está diseñado para agrupar, y los resúmenes pueden aplicarse a cohortes).

6.3. Trabajo Futuro

1. **Afinar el Modelo de Resumen:** Utilizar un modelo de Transformers (BART o T5) específicamente entrenado o afinado (fine-tuned) en español (ej. linorg/beto-summarizer), en lugar de un modelo multilingüe genérico, para mejorar la naturalidad de los resúmenes.
2. **Revisitar el Word Embedding:** En lugar de *entrenar* un modelo (que falló), se deben usar *embeddings pre-entrenados* (como BETO, un BERT en español) para vectorizar las respuestas. Esto permitiría un análisis de similitud (clustering) mucho más robusto en textos cortos.
3. **Análisis de Sentimiento por Tema:** Combinar la extracción de temas (Fase 2) con el análisis de sentimiento (Fase 1) para determinar no solo que la "comunicación" es un tema, sino si el sentimiento *acerca de* la comunicación es positivo o negativo.

7. Referencias Bibliográficas

- Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.
- Lewis, M., Liu, Y., Goyal, N., et al. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv preprint arXiv:1910.13461*.
- VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media.

8. Anexos

8.1. Código Conceptual del Pipeline de Resumen (Transformers)

```
from transformers import pipeline

# 1. Cargar el modelo de resumen (pipeline)
# Se puede usar un modelo en español como 'linorg/beto-summarizer' o uno multilingüe
summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

# 2. Concatenar todas las respuestas de una pregunta (ej. Pregunta 5)
all_responses = " ".join(df['Factores'].dropna().tolist())

# 3. Lógica de Resumen en Lotes (Batches) para GPU
max_chunk_length = 1024 # Límite de tokens del modelo
```

```
batches = [all_responses[i:i + max_chunk_length] for i in range(0, len(all_responses),
max_chunk_length)]

intermediate_summaries = []
for batch in batches:
    summary = summarizer(batch, max_length=150, min_length=30, do_sample=False)
    intermediate_summaries.append(summary[0]['summary_text'])

# 4. Resumen final (resumir los resúmenes)
final_summary_text = " ".join(intermediate_summaries)
final_summary = summarizer(final_summary_text, max_length=250, min_length=50,
do_sample=False)

print(final_summary[0]['summary_text'])
# (Esto produce las salidas guardadas en summarized_answers.csv y factors_summary_5.txt)
```

8.2. Datos Resumidos (Muestra)

Fuente: summarized_answers.csv [cite: summarized_answers.csv]

- **Pregunta:** What is the meaning of cohesion for you in a work team?
- **Resumen:** "Good team integration... All team members know what their work is and that of others and that there is always clear communication. Have a good workflow that everyone has the same direction... Have well-defined roles and build good work environment."
- **Pregunta:** APPOINTMENT A SITUATION THAT YOU CREATE THIS TYPE OF COHESION
- **Resumen:** "When detecting a system failure, gather to know what the fault component may be. When distributing the workload, the work is already done and used as a reference. When the code falls they do not point guilty but collectively seek how to fix the problem."