Problemática: Las relaciones interpersonales son esenciales para nuestro bienestar, pero la mala comunicación puede dañarlas gravemente. Esta se manifiesta como falta de claridad, mala escucha, críticas excesivas y evasión de emociones. Las consecuencias incluyen: Desgaste de la confianza: Se genera sospecha, rencor y miedo. Aumento de conflictos: Desacuerdos, peleas y hasta violencia verbal o emocional. Distanciamiento emocional: Aislamiento, soledad e infelicidad. Deterioro de la relación: Rupturas familiares, distanciamiento de amigos o divorcios. Ejemplo: La crítica hiriente genera resentimiento, daña la autoestima y destruye la confianza.

Importancia de la solución: Una comunicación efectiva es fundamental para relaciones sanas.

Propuesta de solución: Educación en comunicación: Enseñar habilidades desde temprana edad. Empatía y comprensión: Ponerse en el lugar del otro y comprender sus perspectivas. Resolución de conflictos: Estrategias para manejar desacuerdos de manera constructiva. Terapia y asesoramiento: Apoyo profesional para quienes tienen dificultades para comunicarse. Al abordar la mala comunicación, podemos crear relaciones prósperas y llenas de apoyo, amor y conexión.

In [6]:
```
import openai
openai.api_key=""
```

In [8]:
```
#Make the context of our prompt envoirement context = "El trabajo presentado realiza un análisis exhaustivo sob
prompt = "¿Cómo desarrollar herramientas de inteligencia artificial para mejorar la comunicación interpersonal y
```

In [ ]:
```
# Importamos las bibliotecas necesarias
import pandas as pd
import numpy as np
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import
 TfidfVectorizer
from sklearn.cluster
 import KMeans

# Simulamos un conjunto de datos de conversaciones
data = pd.DataFrame({'conversacion': ['Esta conversación es muy positiva', 'Estoy un poco molesto', 'Creo que p
                     'etiqueta': ['positiva', 'negativa', 'neutral', 'negativa']})

# Análisis de sentimiento
sia = SentimentIntensityAnalyzer()
data['sentimiento'] = data['conversacion'].apply(lambda x: sia.polarity_scores(x)['compound'])

# Vectorización de texto
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['conversacion'])

# Clustering de conversaciones
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
data['cluster'] = kmeans.labels_

# Visualización (simulada)
# ... (Aquí podrías usar bibliotecas como matplotlib o seaborn para visualizar los resultados)

print(data)
```