


UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD POLITÉCNICA

DISEÑO DE COMPILADORES

Análisis de Interacciones en Contact Centers con Speech Analytics y Tokenización

Profesor: Sergio Andres Aranda Zeman

Alumno: Lorenzo Carlos Cabrera Villamayor

22 de noviembre del 2024

Funcionalidades Implementadas

1. **Lectura de Transcripción:**
   Se leen las conversaciones desde dos archivos de texto
   - audio_1_a_texto.txt.
   - caso_de_ejemplo_1.txt
   - caso_de_ejemplo_2.txt
   - caso_de_ejemplo_3.txt

2. **Tokenización:**
   Se segmentan las transcripciones en palabras usando expresiones regulares, lo que permite identificar lexemas clave.

3. **Análisis de Sentimiento:**
   Se clasifican las palabras en positivas o negativas según diccionarios predefinidos, sumando su ponderación para determinar el sentimiento general de la conversación.

4. **Verificación del Protocolo:**
   Se asegura que el agente cumple con los elementos básicos del protocolo:
   - Saludo inicial
   - Identificación del cliente
   - Evitar palabras rudas
   - Despedida amable

5. **Generación de Reportes:**
   Se muestra un informe en la consola que detalla:
   - Cumplimiento del protocolo
   - Sentimiento general de la conversación
   - Palabras positivas y negativas detectadas

Decisiones Adoptadas

Tokenización
Se utilizó la biblioteca **re** para tokenizar el texto mediante expresiones regulares, garantizando una separación precisa de palabras.

Arreglos
En la primera versión del código se utilizaron arreglos para guardar las palabras que conforman los diccionarios, esta fue una medida para agilizar el desarrollo del código inicial para conseguir resultados.

Hashing
Finalmente luego de crear un algoritmo base capaz de analizar el texto entre el agente y el cliente busqué alternativas más veloces para la introducción y búsqueda de palabras dentro de los contenedores. Desconocía este dato pero en python, los diccionarios ya están implementados como tablas hash por lo que no necesité implementar hashing para manejar operaciones.

Diccionarios
En reemplazo de los arreglos implementaron diccionarios específicos para palabras positivas, negativas, saludos, identificaciones y despedidas. Esto facilitó el acceso a los datos y otorgó al script una velocidad superior.

Modularización
Por la longitud del código decidí modularizar el script en tres partes:

- **main.py:** Contiene la lógica principal del programa, como el menú y la gestión de las opciones del usuario. Actúa como el controlador del sistema.
- **core.py:** Define las funciones esenciales para el análisis, como la lectura del archivo, tokenización, análisis de sentimiento y verificación del protocolo. Es el núcleo funcional del sistema.
- **dictionaries.py:** Almacena los diccionarios de palabras positivas, negativas y frases clave para el protocolo. Representa la base de datos de términos del sistema.

Con esto se ganó claridad en el código, es posible reutilizar funciones de core.py, facilidad al localizar errores o agregar nuevas funciones.

Código Fuente

main.py

Gestiona el menú principal, permite elegir entre analizar una conversación o agregar palabras a los diccionarios.

```python
import os
import msvcrt
from core import (
    analizar,
    agregar_palabra,
    palabras_positivas,
    palabras_negativas,
    saludos,
    identificacion,
    despedidas,
)

# Menu
while True:
    os.system("cls")
    print("1 - Analizar\n2 - Agregar palabras \n0 - Salir")
    entrada = input("Elija una opción: ").strip()
    if entrada == "1":
        while True:
            os.system("cls")
            print(
                "1 - Analizar 'audio_a_texto'\n2 - Analizar
'caso_de_ejemplo'\n3 - Analizar caso: cliente molesto\n4 - Analizar caso:
agente alterado\n0 - Volver"
            )
            eleccion = input("Elija una opción: ").strip()
            if eleccion == "1":
                nombre_archivo = "audio_1_a_texto.txt"
                analizar(nombre_archivo)
            elif eleccion == "2":
                nombre_archivo = "caso_de_ejemplo_1.txt"
                analizar(nombre_archivo)
            elif eleccion == "3":
                nombre_archivo = "caso_de_ejemplo_2.txt"
                analizar(nombre_archivo)
            elif eleccion == "4":
                nombre_archivo = "caso_de_ejemplo_3.txt"
                analizar(nombre_archivo)
            elif eleccion == "0":
```

```
                os.system("cls")
                break

    elif entrada == "2":
        while True:
            os.system("cls")
            print("A qué conjunto de palabras desea agregar?")
            print(
                "1 - Palabras positivas\n2 - Palabras negativas\n3 -
Saludos\n4 - Identificación del cliente\n5 - Despedidas\n0 - Volver"
            )
            eleccion = input("Elija una opción: ").strip()
            if eleccion == "1":
                while True:
                    palabra = input("Inserte una palabra: ").strip()
                    valor = input("Asigne un valor: ").strip()
                    if valor.isdigit() and not palabra.isdigit() and palabra
!= "":
                        agregar_palabra(palabras_positivas, palabra,
int(valor))
                        print(f"Palabra '{palabra}' añadida con valor
{valor}.")
                        print("Presione una tecla para continuar...")
                        msvcrt.getch()
                        break
            elif eleccion == "2":
                while True:
                    palabra = input("Inserte una palabra: ").strip()
                    valor = input("Asigne un valor: ").strip()
                    if valor.isdigit() and not palabra.isdigit() and palabra
!= "":
                        agregar_palabra(palabras_negativas, palabra,
int(valor))
                        print(f"Palabra '{palabra}' añadida con valor
{valor}.")
                        print("Presione una tecla para continuar...")
                        msvcrt.getch()
                        break
            elif eleccion == "3":
                while True:
                    palabra = input("Inserte una palabra: ").strip()
                    valor = ""
                    if not palabra.isdigit() and palabra != "":
                        agregar_palabra(saludos, palabra, valor)
                        print(f"Palabra '{palabra}' añadida.")
                        print("Presione una tecla para continuar...")
                        msvcrt.getch()
                        break
```

```
        elif eleccion == "4":
            while True:
                palabra = input("Inserte una palabra: ").strip()
                valor = ""
                if not palabra.isdigit() and palabra != "":
                    agregar_palabra(identificacion, palabra, valor)
                    print(f"Palabra '{palabra}' añadida.")
                    print("Presione una tecla para continuar...")
                    msvcrt.getch()
                    break
        elif eleccion == "5":
            while True:
                palabra = input("Inserte una palabra: ").strip()
                valor = ""
                if not palabra.isdigit() and palabra != "":
                    agregar_palabra(despedidas, palabra, valor)
                    print(f"Palabra '{palabra}' añadida.")
                    print("Presione una tecla para continuar...")
                    msvcrt.getch()
                    break
        elif eleccion == "0":
            os.system("cls")
            break
    elif entrada == "0":
        os.system("cls")
        break
```

core.py

Contiene las funciones principales para la lectura del archivo, tokenización, análisis de palabras clave, verificación del protocolo y generación del reporte.

```python
import os
import re
import msvcrt
from dictionaries import (
    palabras_positivas,
    palabras_negativas,
    saludos,
    identificacion,
    despedidas,
)


class Color:
    VERDE = "\033[32m"
    AMARILLO = "\033[33m"
    RESTABLECER = "\033[0m"


# 1. Leer el archivo de texto
def leer_transcripcion(nombre_archivo):
    ruta_archivo = os.path.join("texto", nombre_archivo)
    with open(ruta_archivo, "r", encoding="utf-8") as archivo:
        return archivo.readlines()


# 2. Separar diálogos por roles
def separar_dialogos(texto):
    agente = []
    cliente = []
    for linea in texto:
        if linea.startswith("Agente:"):
            agente.append(linea.replace("Agente:", "").strip())
        elif linea.startswith("Cliente:"):
            cliente.append(linea.replace("Cliente:", "").strip())
    return agente, cliente


# 3. Tokenizar las frases
def tokenizar(frases):
    tokens = []
    for frase in frases:
```

```
        # Separar palabras por espacios y remover signos de puntuación
        tokens += re.findall(r"\b\w+\b", frase.lower())
    return tokens


# 4. Analizar palabras clave
def analizar_palabras_clave(tokens):
    palabras_detectadas = {"positivas": [], "negativas": []}
    ponderacion_total = 0

    for token in tokens:
        if token in palabras_positivas:
            peso = palabras_positivas[token]
            palabras_detectadas["positivas"].append({token: peso})
            ponderacion_total += peso
        elif token in palabras_negativas:
            peso = palabras_negativas[token]
            palabras_detectadas["negativas"].append({token: peso})
            ponderacion_total += peso

    return palabras_detectadas, ponderacion_total


# 5. Verificar protocolo
def verificar_protocolo(agente_dialogos):
    protocolo = {
        "saludo": False,
        "identificacion_cliente": False,
        "despedida_amable": False,
        "palabras_positivas": [],
        "palabras_negativas": [],
        "ponderacion_sentimiento": 0,
    }

    # Calcular segmentos del diálogo
    inicio = int(len(agente_dialogos) * 0.4)  # 40% inicial
    final = int(len(agente_dialogos) * 0.7)  # 30% final
    inicio_dialogo = agente_dialogos[:inicio]
    final_dialogo = agente_dialogos[final:]

    # Tokenizar segmentos
    tokens_inicio = tokenizar(inicio_dialogo)
    tokens_final = tokenizar(final_dialogo)
    tokens_completos = tokenizar(agente_dialogos)

    # Verificar saludo e identificación en el 40% inicial
    protocolo["saludo"] = any(palabra in tokens_inicio for palabra in
saludos)
```

```
    protocolo["identificacion_cliente"] = any(
        frase in " ".join(tokens_inicio) for frase in identificacion
    )

    # Verificar despedida amable en el 30% final
    protocolo["despedida_amable"] = any(
        frase in " ".join(tokens_final) for frase in despedidas
    )

    # Analizar palabras positivas y negativas
    palabras_detectadas, ponderacion_total =
analizar_palabras_clave(tokens_completos)
    protocolo["palabras_positivas"] = palabras_detectadas["positivas"]
    protocolo["palabras_negativas"] = palabras_detectadas["negativas"]
    protocolo["ponderacion_sentimiento"] = ponderacion_total

    return protocolo


# 6. Obtener palabra más positiva
def obtener_mas_positiva(palabras_positivas):
    if not palabras_positivas:
        return None

    mas_positiva = max(
        palabras_positivas, key=lambda x: list(x.values())[0], default=None
    )
    return mas_positiva


# 7. Obtener palabra más negativa
def obtener_mas_negativa(palabras_negativas):
    if not palabras_negativas:
        return None

    mas_negativa = min(
        palabras_negativas, key=lambda x: list(x.values())[0], default=None
    )
    return mas_negativa


# 8. Impresión del protocolo de atención
def imprimir_protocolo_atencion(protocolo_agente):
    print("\n\tProtocolo de Atención")
    print(
        "Fase de saludo:",
        (
            (
```

```
                (Color.VERDE + "OK")
                if protocolo_agente.get("saludo")
                else (Color.AMARILLO + "Faltante")
            )
            + Color.RESTABLECER
        ),
    )
    print(
        "Identificación del cliente:",
        (
            (
                (Color.VERDE + "OK")
                if protocolo_agente.get("identificacion_cliente")
                else (Color.AMARILLO + "Faltante")
            )
            + Color.RESTABLECER
        ),
    )
    print(
        "Uso de palabras rudas: ",
        (
            (
                (Color.AMARILLO + "Detectadas")
                if protocolo_agente.get("palabras_negativas")
                else (Color.VERDE + "Ninguna detectada")
            )
            + Color.RESTABLECER
        ),
    )
    print(
        "Despedida amable:",
        (
            (
                (Color.VERDE + "OK")
                if protocolo_agente.get("despedida_amable")
                else (Color.AMARILLO + "Faltante")
            )
            + Color.RESTABLECER
        ),
    )


# 9. Impresión
def imprimir_deteccion_sentimiento(
    sentimiento_general,
    cantidad_palabras_positivas,
    palabra_mas_positiva,
    cantidad_palabras_negativas,
```

```
    palabra_mas_negativa,
):
    print("\n\tDetección de Sentimiento")
    print("Sentimiento general: ", sentimiento_general + Color.RESTABLECER)
    print(
        "Palabras positivas: ",
        Color.VERDE + f"{cantidad_palabras_positivas}" + Color.RESTABLECER,
    )
    if palabra_mas_positiva:
        palabra, peso = list(palabra_mas_positiva.items())[0]
        print(
            f"Palabra más positiva: {Color.VERDE}{palabra},
+{peso}{Color.RESTABLECER}"
        )
    else:
        print(f"Palabra más positiva: {Color.AMARILLO}No
detectada{Color.RESTABLECER}")
    print(
        "Palabras negativas: ",
        Color.AMARILLO + f"{cantidad_palabras_negativas}" +
Color.RESTABLECER,
    )
    if palabra_mas_negativa:
        palabra, peso = list(palabra_mas_negativa.items())[0]
        print(
            f"Palabra más negativa: {Color.AMARILLO}{palabra},
{peso}{Color.RESTABLECER}\n"
        )
    else:
        print(f"Palabra más negativa: {Color.VERDE}No
detectada{Color.RESTABLECER}\n")


# 10. Agregar palabras a los dicionarios
def agregar_palabra(diccionario, palabra, valor):
    diccionario[palabra.lower()] = valor


# 11. Analizar texto
def analizar(nombre_archivo):
    texto = leer_transcripcion(nombre_archivo)
    agente_dialogos, cliente_dialogos = separar_dialogos(texto)

    # Análisis del agente
    protocolo_agente = verificar_protocolo(agente_dialogos)

    # Análisis del cliente
    tokens_cliente = tokenizar(cliente_dialogos)
```

```
    sentimiento_cliente, ponderacion_cliente =
analizar_palabras_clave(tokens_cliente)

    # Análisis del sentimiento general de la conversación
    ponderacion_general = (
        protocolo_agente["ponderacion_sentimiento"] + ponderacion_cliente
    )
    if ponderacion_general > 0:
        sentimiento_general = Color.VERDE + f"Positivo
(+{ponderacion_general})"
    elif ponderacion_general < 0:
        sentimiento_general = Color.AMARILLO + f"Negativo
({ponderacion_general})"
    else:
        sentimiento_general = f"Neutral (0)"

    # Cálculo de la cantidad de palabras positivas y negativas recogidas
    cantidad_palabras_positivas = len(sentimiento_cliente["positivas"]) +
len(
        protocolo_agente["palabras_positivas"]
    )
    cantidad_palabras_negativas = len(sentimiento_cliente["negativas"]) +
len(
        protocolo_agente["palabras_negativas"]
    )

    # Busqueda de palabra más positiva y más negativa
    palabra_mas_positiva = obtener_mas_positiva(
        protocolo_agente["palabras_positivas"] +
sentimiento_cliente["positivas"]
    )
    palabra_mas_negativa = obtener_mas_negativa(
        protocolo_agente["palabras_negativas"] +
sentimiento_cliente["negativas"]
    )

    # Impresión de resultados
    os.system("cls")
    # PROTOCOLO DE ATENCIÓN
    imprimir_protocolo_atencion(protocolo_agente)

    # DETECCIÓN DE SENTIMIENTO
    imprimir_deteccion_sentimiento(
        sentimiento_general,
        cantidad_palabras_positivas,
        palabra_mas_positiva,
        cantidad_palabras_negativas,
        palabra_mas_negativa,
```

```
)

print("Presione una tecla para continuar...")
msvcrt.getch()
os.system("cls")
```

diccionarios.py

Define los diccionarios de palabras positivas, negativas, saludos, identificaciones y despedidas.

```python
palabras_positivas = {
    "excelente": 50,
    "excelentes": 50,
    "amable": 49,
    "amables": 49,
    "profesional": 48,
    "profesionales": 48,
    "rápido": 47,
    "rápidos": 47,
    "rápida": 47,
    "rápidas": 47,
    "eficiente": 46,
    "eficientes": 46,
    "maravilloso": 45,
    "maravillosos": 45,
    "maravillosa": 45,
    "maravillosas": 45,
    "perfecto": 44,
    "perfectos": 44,
    "perfecta": 44,
    "perfectas": 44,
    "resuelto": 43,
    "resueltos": 43,
    "resuelta": 43,
    "resueltas": 43,
    "increíble": 42,
    "increíbles": 42,
    "oportuno": 41,
    "oportunos": 41,
    "oportuna": 41,
    "oportunas": 41,
    "útil": 40,
    "útiles": 40,
    "satisfactorio": 39,
    "satisfactorios": 39,
    "satisfactoria": 39,
    "satisfactorias": 39,
    "agradable": 38,
    "agradables": 38,
    "cómodo": 37,
    "cómodos": 37,
```

```
"cómoda": 37,
"cómodas": 37,
"claro": 36,
"claros": 36,
"clara": 36,
"claras": 36,
"confiable": 35,
"confiables": 35,
"atento": 34,
"atentos": 34,
"atenta": 34,
"atentas": 34,
"educado": 33,
"educados": 33,
"educada": 33,
"educadas": 33,
"bueno": 32,
"buenos": 32,
"buena": 32,
"buenas": 32,
"simple": 31,
"simples": 31,
"eficaz": 30,
"eficaces": 30,
"respetuoso": 29,
"respetuosos": 29,
"respetuosa": 29,
"respetuosas": 29,
"honesto": 28,
"honestos": 28,
"honesta": 28,
"honestas": 28,
"dedicado": 27,
"dedicados": 27,
"dedicada": 27,
"dedicadas": 27,
"puntual": 26,
"puntuales": 26,
"proactivo": 25,
"proactivos": 25,
"proactiva": 25,
"proactivas": 25,
"servicial": 24,
"serviciales": 24,
"extraordinario": 23,
"extraordinarios": 23,
"extraordinaria": 23,
"extraordinarias": 23,
```

```
"encantador": 22,
"encantadores": 22,
"encantadora": 22,
"encantadoras": 22,
"impresionante": 21,
"impresionantes": 21,
"espectacular": 20,
"espectaculares": 20,
"destacado": 19,
"destacados": 19,
"destacada": 19,
"destacadas": 19,
"fantástico": 18,
"fantásticos": 18,
"fantástica": 18,
"fantásticas": 18,
"excelencia": 17,
"privilegiado": 16,
"privilegiados": 16,
"privilegiada": 16,
"privilegiadas": 16,
"innovador": 15,
"innovadores": 15,
"innovadora": 15,
"innovadoras": 15,
"valioso": 14,
"valiosos": 14,
"valiosa": 14,
"valiosas": 14,
"insuperable": 13,
"insuperables": 13,
"magnífico": 12,
"magníficos": 12,
"magnífica": 12,
"magníficas": 12,
"glorioso": 11,
"gloriosos": 11,
"gloriosa": 11,
"gloriosas": 11,
"concreto": 10,
"concretos": 10,
"concreta": 10,
"concretas": 10,
"seguro": 8,
"seguros": 8,
"segura": 8,
"seguras": 8,
"apropiado": 7,
```

```
    "apropiados": 7,
    "apropiada": 7,
    "apropiadas": 7,
    "justo": 6,
    "justos": 6,
    "justa": 6,
    "justas": 6,
    "comprensible": 5,
    "comprensibles": 5,
    "práctico": 4,
    "prácticos": 4,
    "práctica": 4,
    "prácticas": 4,
    "amigable": 3,
    "amigables": 3,
    "responsable": 2,
    "responsables": 2,
    "conveniente": 1,
    "convenientes": 1,
}

palabras_negativas = {
    "horrible": -50,
    "horribles": -50,
    "estafa": -49,
    "estafas": -49,
    "mentiroso": -48,
    "mentirosos": -48,
    "mentirosa": -48,
    "mentirosas": -48,
    "vergüenza": -47,
    "vergüenzas": -47,
    "nefasto": -46,
    "nefastos": -46,
    "nefasta": -46,
    "nefastas": -46,
    "patético": -45,
    "patéticos": -45,
    "patética": -45,
    "patéticas": -45,
    "ineficiente": -44,
    "ineficientes": -44,
    "lento": -43,
    "lentos": -43,
    "lenta": -43,
    "lentas": -43,
    "confuso": -42,
    "confusos": -42,
```

```
"confusa": -42,
"confusas": -42,
"dudoso": -41,
"dudosos": -41,
"dudosa": -41,
"dudosas": -41,
"abusivo": -40,
"abusivos": -40,
"abusiva": -40,
"abusivas": -40,
"desastroso": -39,
"desastrosos": -39,
"desastrosa": -39,
"desastrosas": -39,
"engañoso": -38,
"engañosos": -38,
"engañosa": -38,
"engañosas": -38,
"pésimo": -37,
"pésimos": -37,
"pésima": -37,
"pésimas": -37,
"insufrible": -36,
"insufribles": -36,
"defectuoso": -35,
"defectuosos": -35,
"defectuosa": -35,
"defectuosas": -35,
"irresponsable": -34,
"irresponsables": -34,
"desagradable": -33,
"desagradables": -33,
"tóxico": -32,
"tóxicos": -32,
"tóxica": -32,
"tóxicas": -32,
"incompetente": -31,
"incompetentes": -31,
"problemático": -30,
"problemáticos": -30,
"problemática": -30,
"problemáticas": -30,
"incómodo": -29,
"incómodos": -29,
"incómoda": -29,
"incómodas": -29,
"fallido": -28,
"fallidos": -28,
```

```
"frustrante": -27,
"frustrantes": -27,
"complicado": -26,
"complicados": -26,
"complicada": -26,
"complicadas": -26,
"desorganizado": -25,
"desorganizados": -25,
"desorganizada": -25,
"desorganizadas": -25,
"ineficaz": -24,
"ineficaces": -24,
"injusto": -23,
"injustos": -23,
"injusta": -23,
"injustas": -23,
"irrespetuoso": -22,
"irrespetuosos": -22,
"irrespetuosa": -22,
"irrespetuosas": -22,
"decepcionante": -21,
"decepcionantes": -21,
"intolerable": -20,
"intolerables": -20,
"precario": -19,
"precarios": -19,
"precaria": -19,
"precarias": -19,
"inepto": -18,
"ineptos": -18,
"inepta": -18,
"ineptas": -18,
"mediocre": -17,
"mediocres": -17,
"caótico": -16,
"caóticos": -16,
"caótica": -16,
"caóticas": -16,
"obsoleto": -15,
"obsoletos": -15,
"obsoleta": -15,
"obsoletas": -15,
"retrasado": -14,
"retrasados": -14,
"retrasada": -14,
"retrasadas": -14,
"estresante": -13,
"estresantes": -13,
```

```
    "repetitivo": -12,
    "repetitivos": -12,
    "repetitiva": -12,
    "repetitivas": -12,
    "pesado": -11,
    "pesados": -11,
    "pesada": -11,
    "pesadas": -11,
    "agotador": -10,
    "agotadores": -10,
    "agotadora": -10,
    "agotadoras": -10,
    "impuntual": -9,
    "impuntuales": -9,
    "difícil": -8,
    "difíciles": -8,
    "ambiguo": -7,
    "ambiguos": -7,
    "ambigua": -7,
    "ambiguas": -7,
    "forzado": -6,
    "forzados": -6,
    "forzada": -6,
    "forzadas": -6,
    "irritante": -5,
    "irritantes": -5,
    "trivial": -4,
    "triviales": -4,
    "molesto": -3,
    "molestos": -3,
    "molesta": -3,
    "molestas": -3,
    "innecesario": -2,
    "innecesarios": -2,
    "innecesaria": -2,
    "innecesarias": -2,
    "deshonesto": -1,
    "deshonestos": -1,
    "deshonesta": -1,
    "deshonestas": -1,
}

saludos = {
    "hola": "",
    "buenas": "",
    "buen día": "",
    "buenas tardes": "",
    "buenas noches": "",
```

```
    "bienvenido": "",
    "qué tal": "",
    "cómo está": "",
    "cómo estás": "",
    "cómo se encuentra": "",
    "cómo te encuentras": "",
    "un placer atenderte": "",
    "es un gusto atenderle": "",
    "bienvenido a nuestro servicio": "",
    "es un placer ayudarle": "",
    "encantado de atenderle": "",
    "qué gusto escucharle": "",
    "saludos cordiales": "",
    "gracias por comunicarse": "",
    "gracias por llamar": "",
    "un cordial saludo": "",
    "le damos la bienvenida": "",
    "qué gusto saludarle": "",
    "buen día, ¿cómo puedo ayudarle?": "",
    "estamos aquí para ayudarle": "",
    "gracias por elegirnos": "",
    "hola, buen día, ¿cómo está?": "",
    "espero se encuentre bien": "",
    "mucho gusto, estoy para ayudarle": "",
    "saludos y gracias por contactarnos": "",
    "le damos un caluroso saludo": "",
    "bienvenido al servicio de atención": "",
    "cómo puedo asistirle hoy": "",
    "le saludo cordialmente": "",
    "gracias por ponerse en contacto": "",
    "muy buenas, ¿en qué puedo servirle?": "",
    "gracias por darnos la oportunidad de ayudarle": "",
    "bienvenido, estamos a su disposición": "",
    "muchas gracias por contactarnos": "",
    "hola, ¿cómo le podemos apoyar hoy?": "",
    "es un gusto atender su llamada": "",
    "gracias por tomarse el tiempo de llamarnos": "",
    "le agradecemos su comunicación": "",
    "buen día, estoy aquí para resolver sus dudas": "",
    "muy buenas tardes, ¿en qué puedo ayudarle?": "",
    "buenas noches, gracias por su tiempo": "",
    "muchas gracias por elegirnos": "",
    "hola, gracias por confiar en nosotros": "",
    "gracias por contactarnos, estamos para servirle": "",
    "buen día, ¿cómo puedo atenderle?": "",
}
```

```
identificacion = {
    "tu nombre": "",
    "cómo se llama": "",
    "su nombre por favor": "",
    "con quién tengo el gusto": "",
    "me brinda su nombre": "",
    "puedo saber su nombre": "",
    "a nombre de quién registro esto": "",
    "me confirma su nombre": "",
    "su identificación por favor": "",
    "puedo verificar su nombre": "",
    "me indica su nombre completo": "",
    "cómo le registro en nuestro sistema": "",
    "me puede confirmar su identidad": "",
    "nombre completo, por favor": "",
    "a quién tengo el honor de atender": "",
    "por favor, su nombre completo": "",
    "quién me está hablando": "",
    "me facilita su nombre": "",
    "me confirma su identidad, por favor": "",
    "puede proporcionarme su nombre": "",
    "necesito confirmar su nombre": "",
    "puede decirme su nombre completo": "",
    "cómo prefiero dirigirme a usted": "",
    "a quién debo registrar esta solicitud": "",
    "por favor, confírmeme su identidad": "",
    "puede indicarme quién está llamando": "",
    "necesito registrar su nombre completo": "",
    "me confirma quién está al teléfono": "",
    "su nombre completo para el registro, por favor": "",
    "cómo figura en nuestro sistema": "",
    "puedo anotar su nombre, por favor": "",
    "es usted quien figura en el registro": "",
    "a nombre de quién dejo la solicitud": "",
    "puede ayudarme con su nombre": "",
    "por favor, dígame cómo puedo identificarle": "",
    "necesito validar su nombre para el trámite": "",
    "por favor, me brinda sus datos": "",
    "su nombre es necesario para continuar": "",
    "puede verificarme su identidad, por favor": "",
    "me podría proporcionar su identidad": "",
    "cómo desea que le registre en el sistema": "",
    "puedo saber a quién estoy atendiendo": "",
    "indíqueme su nombre, por favor": "",
    "sería tan amable de confirmarme su identidad": "",
    "necesito saber quién está al teléfono": "",
    "cómo prefiero llamarle": "",
    "indíqueme su identidad, por favor": "",
```

```
    "puede indicarme su nombre y apellido": "",
}


despedidas = {
    "gracias por su tiempo": "",
    "que tenga un buen día": "",
    "hasta luego": "",
    "que esté bien": "",
    "gracias por llamar": "",
    "estamos para servirle": "",
    "cuídese": "",
    "esperamos su llamada pronto": "",
    "ha sido un placer atenderle": "",
    "muchas gracias por comunicarse con nosotros": "",
    "le deseamos un excelente día": "",
    "gracias por preferirnos": "",
    "que tenga una excelente tarde": "",
    "quedamos atentos a su próxima llamada": "",
    "que tenga una noche tranquila": "",
    "si necesita algo más, estamos aquí para usted": "",
    "no dude en llamarnos de nuevo": "",
    "gracias por confiar en nosotros": "",
    "hasta la próxima": "",
    "que tenga un excelente día, hasta luego": "",
    "gracias por contactarnos, le deseamos lo mejor": "",
    "un cordial saludo, hasta luego": "",
    "fue un gusto atenderle, hasta pronto": "",
    "gracias por su confianza": "",
    "nos despedimos cordialmente": "",
    "le agradecemos su preferencia": "",
    "quedamos atentos a cualquier necesidad": "",
    "estaremos aquí cuando lo necesite": "",
    "gracias por permitirnos ayudarle": "",
    "le deseamos todo lo mejor": "",
    "que pase un excelente día": "",
    "le agradecemos profundamente su tiempo": "",
    "ha sido un gusto apoyarle": "",
    "nos despedimos y le deseamos lo mejor": "",
    "que pase una noche excelente": "",
    "gracias por darnos la oportunidad de ayudarle": "",
    "estaremos esperando su próxima llamada": "",
    "gracias por llamarnos, fue un placer": "",
    "que su día continúe de maravilla": "",
    "le deseamos una semana fantástica": "",
    "hasta pronto, y gracias por contactarnos": "",
    "que pase una tarde excelente": "",
    "gracias por confiar en nuestros servicios": "",
```

```
    "nos despedimos con un cordial saludo": "",
    "fue un honor atenderle hoy": "",
    "nos despedimos y quedamos a su disposición": "",
    "gracias por elegirnos, hasta luego": "",
    "esperamos volver a atenderle pronto": "",
    "que tenga un día lleno de éxitos": "",
    "gracias por su preferencia, le esperamos": "",
}
```

Resultado para un Caso de Ejemplo

Entrada:
Archivo de texto con el siguiente contenido:

```
Unset
Agente: Hola, bienvenido a Atención al Cliente. ¿Con quién tengo el gusto de hablar?
Cliente: Buenas, mi nombre es Juan Pérez.
Agente: ¿En qué puedo ayudarle hoy?
Cliente: Tengo un problema con mi factura, el monto es incorrecto.
Agente: Vamos a revisar eso. Por favor, deme un momento.
Cliente: Gracias.
Agente: Ya revisé su caso. Todo está solucionado. ¿Puedo ayudarle en algo más?
Cliente: No, eso es todo.
Agente: Muchas gracias por su tiempo. Que tenga un buen día.
```

Salida esperada:

```
Python
Protocolo de Atención:
Fase de saludo: OK
Identificación del cliente: OK
Uso de palabras rudas: Ninguna detectada
Despedida amable: OK

Detección de Sentimiento:
Sentimiento general: Neutral (0)
Palabras positivas: 0
Palabra más positiva: No detectada
Palabras negativas: 0
Palabra más negativa: No detectada
```

Observaciones y Situaciones no Resueltas

Diccionario de la lengua española:
A fin de focalizar la atención en las palabras buscadas opté por no tener un diccionario general que indique si es o no una palabra válida, elegí reducir el costo de búsqueda centrándome en cada uno de los tipos de palabras que afectan directamente el reporte final

Conversión de Audio a Texto:
El sistema actual supone que las transcripciones ya están en formato de texto. No se ha implementado la conversión de audio a texto, que sería necesaria para cumplir con la totalidad del enunciado.

Expansión de Diccionarios:
Los diccionarios de palabras positivas, negativas y de protocolo pueden ampliarse según el contexto específico del contact center.

Detección de Sarcasmo o Contexto:
El análisis de sentimiento se basa solo en palabras individuales y no considera el contexto o el tono de la conversación, lo que podría afectar la precisión en algunos casos.