Avance 2. Ingeniería de características

Ángel Efraín Luna Martínez - A01795486
Francisco Salvador Hernández Pérez - A01795486
Iker Bring Anaya - A01795270

Contenido

Introducción

El presente documento corresponde a la fase de Ingeniería de Características (Feature Engineering) dentro de la metodología CRISP-ML. A diferencia de los proyectos tradicionales basados en datos tabulares, este proyecto tiene como objetivo desarrollar una aplicación que reciba archivos de audio provenientes de diversas plataformas de mensajería (Google Chat, WhatsApp, Teams, Telegram y Messenger), los procese y retorne el texto transcrito junto con un identificador único de seguimiento.

Dado que los datos de entrada son señales de audio y no variables numéricas o categóricas, el uso de una libreta de Jupyter para la preparación y análisis exploratorio de datos no resulta pertinente. En su lugar, se presenta este análisis técnico-documental que describe los formatos, códecs, y flujos de recuperación de los audios, justificando los procesos de normalización y conversión necesarios para su posterior uso por motores de reconocimiento de voz (Speech-to-Text, STT). De esta forma, el documento cumple el mismo propósito analítico que una libreta, pero adaptado a un entorno de procesamiento de audio.

Contextualización dentro de la Ingeniería de Características

La ingeniería de características en este proyecto implica convertir los audios crudos en representaciones numéricas y textuales útiles. Las operaciones aplicadas incluyen la normalización del audio, la reducción de ruido, la estandarización de formatos y la extracción de características acústicas. Estos procesos cumplen un rol equivalente a la codificación, escalamiento o selección de variables en datasets tradicionales, garantizando compatibilidad, reproducibilidad y eficiencia durante la fase de transcripción.

Formato audio proveedores

Google Chat

Documentación

La documentación oficial de Google explica que en Chat los mensajes pueden incluir adjuntos que contienen datos binarios, incluyendo archivos de audio. Estos adjuntos se acceden a través de la propiedad attachments de un mensaje, la cual expone tanto el tipo de contenido (contentType) como un resourceName para descargar el archivo usando el endpoint media.download.

Formato de las notas de voz

Al explorar la API, las notas de voz se envían como adjuntos en un mensaje. El campo attachments[].contentType indica el formato real.
Se realizó un pequeño experimento donde configuramos un espacio y se creó una nota de voz. Al inspeccionar el mensaje de voz con la API se obtuvo:

- contentName: nombre de archivo con extensión .m4a (ejemplo: *UserRecording_1759465671370.m4a*)

- contentType: audio/mpeg


```
"contentName": "UserRecording_1759465671370.m4a",
"contentType": "audio/mpeg",
"attachmentDataRef": {
    "resourceName":
        "ClxzcGFjZXMvQUFRQW1vT2ZrN1EvbWVzc2FnZXMvC
        obWVudHMvQUFUVWYtTHllX3VFTHlhT1NDa0VIR0tDc
},
```


Esto confirma que Google Chat manda las notas de voz en contenedor M4A (MPEG-4 Audio) con códec AAC (verificable con herramientas como ffprobe).

Cómo se recupera el archivo

Se usó la API:

GET https://chat.googleapis.com/v1/spaces/{space}/messages?pageSize=20&orderBy=createTime desc

También se puede buscar directamente el mensaje en spaces/{space}/messages/{message}.

Dentro del objeto attachments[] aparece el contentType y el attachmentDataRef.resourceName. Este último se utiliza con el endpoint media.download:

GET https://chat.googleapis.com/v1/media/{resourceName}?alt=media

(con un token válido en Authorization: Bearer). El binario resultante es el .m4a original enviado por el usuario.

Implicaciones para STT

Se recibirá un .m4a (audio/mpeg, AAC). Muchos motores de STT lo aceptan directamente, pero si no, conviene convertirlo a WAV PCM mono antes de transcribir.

WhatsApp (Web)

Documentación

La documentación oficial de la Cloud API de WhatsApp indica que los mensajes de audio llegan al webhook como un objeto audio que incluye un id. Ese identificador se utiliza para descargar el archivo real con el endpoint de media. En la metadata del webhook aparece el mime_type que define el formato del audio recibido.

Formato de las notas de voz

WhatsApp graba y envía las notas de voz en formato OGG con códec Opus, en un solo canal (mono).

- En dispositivos móviles, los archivos se guardan con extensión .opus, pero en realidad son contenedores OGG con flujo Opus.

- En WhatsApp Web, el cliente no recodifica, simplemente retransmite el archivo generado en el móvil. Al descargar desde el navegador se obtiene un archivo .ogg con Opus.

Cómo se recupera el archivo

Webhook → objeto audio con id.

Descarga vía API con:

```
GET /v1/media/{media-id}
```

En la metadata:

```
"mime_type": "audio/ogg; codecs=opus"
```

Formatos soportados en general por la API: AAC, AMR, MP3, M4A y OGG/Opus.

Para notas de voz nativas el formato enviado y recibido es siempre OGG/Opus (mono).

Implicaciones para STT

El audio se recibe en OGG/Opus mono. Muchos motores de STT soportan Opus, pero en se podria decodificar a WAV PCM antes de transcribir.

Limitación: tamaño máximo de archivo 16 MB.

Microsoft Teams

Documentación

La documentación oficial de Microsoft explica que los audios en Teams se gestionan como archivos adjuntos o grabaciones y se almacenan en OneDrive/SharePoint. Se pueden recuperar a través de Microsoft Graph API, ya sea como driveItem o como chatMessage/attachment. Además, las grabaciones de reuniones se guardan en .mp4 y, si está activada la transcripción, se generan subtítulos .vtt.

Formato de las notas de voz

- **Clips de audio en chat (voice clips):** enviados como .mp4 (aunque contengan solo audio).
- **Reuniones y llamadas grabadas:** archivos .mp4 (video H.264 + audio AAC).
- **Buzón de voz (Cloud Voicemail):** correo en Exchange con archivo de audio adjunto.
- **Adjuntos genéricos:** Teams permite enviar archivos de audio en su formato original (MP3, WAV, M4A, WMA).

Cómo se recupera el archivo

- **Clips de chat:** descargables desde la UI como .mp4; vía API en chatMessage/attachment/$value.
- **Reuniones:** almacenadas en OneDrive/SharePoint, accesibles vía Graph como driveItem/$value.
- **Voicemail:** entregado como attachment en un correo de Exchange.
- **Adjuntos normales:** permanecen en el formato original y se acceden con Graph.

Implicaciones para STT

La mayoría de los clips y grabaciones llegan en .mp4 con audio AAC, que suele ser compatible con motores de STT. Sin embargo, para máxima interoperabilidad puede convertirse a **WAV PCM**.

Telegram

Documentación

La documentación oficial de la Telegram Bot API explica que las notas de voz se representan mediante el objeto `voice`. Este objeto contiene campos como `file_id`, `mime_type` y `duration`. Con el `file_id` se utiliza el método `getFile`, que devuelve un objeto con el campo `file_path`. Ese valor permite construir la URL para descargar el archivo de audio almacenado en los servidores de Telegram. La especificación del método `sendVoice` también establece que los archivos enviados como notas de voz deben ser de tipo `audio/ogg`, confirmando que el contenedor utilizado es OGG. Adicionalmente, la librería oficial TDLib (Telegram Database Library) documenta que los *voice notes* deben codificarse en Opus, dentro de un contenedor OGG y con un solo canal de audio.

Public Fields

object_ptr< InputFile >	**voice_note_**
	Voice note to be sent. The voice note must be encoded with the Opus codec and stored inside an OGG container with a single audio channel, or be in MP3 or M4A format as regular audio.

Formato de las notas de voz

El análisis de la API y de la documentación confirma que las notas de voz en Telegram se transmiten bajo un esquema estandarizado:

- **Contenedor:** OGG
- **Códec:** Opus (siempre mono)
- **MIME**: audio/ogg
- **Extensión:** .ogg (aunque muchos clientes lo etiquetan como “nota de voz” o *voice message*)

  Este formato es exclusivo para los mensajes de voz. Otros audios que se envían como documentos mantienen el formato original en el que fueron cargados (por ejemplo, MP3, WAV o M4A).

Cómo se recupera el archivo

Cuando llega un mensaje con nota de voz, la API retorna un objeto `voice` con:

- `file_id` → identificador único.
- `mime_type` → que corresponde a `audio/ogg`.

El flujo para obtener el archivo es:

1.Invocar el método `getFile` con el `file_id`:

`https://api.telegram.org/bot<token>/getFile?file_id=<FILE_ID>`

2.La respuesta incluye un `file_path`.

3.Con el `file_path`, descargar el archivo:

`https://api.telegram.org/file/bot<token>/<file_path>`

4.El resultado es un archivo `.ogg` con flujo Opus mono.

Implicaciones para STT

El formato estándar de notas de voz en Telegram es OGG/Opus mono. Algunos motores de STT son compatibles con este formato, pero, para maximizar interoperabilidad y evitar problemas de decodificación, se puede podria usar WAV PCM mono antes de iniciar el proceso de transcripción.

Messenger

Documentación

La documentación de la Messenger Platform establece que los mensajes pueden incluir archivos adjuntos de tipo audio, los cuales se entregan en el objeto attachments con un payload.url. Esa URL es temporal y requiere un Page Access Token válido para descargar el archivo. La documentación de la Attachment Upload API especifica que los formatos de audio soportados en Messenger incluyen AAC, M4A, WAV y MP4, lo que refleja la manera en que los clientes móviles graban y envían notas de voz.

Adicionalmente, experiencias de desarrolladores que han procesado estos adjuntos muestran que las notas de voz descargadas desde Messenger se reciben como archivos M4A (contenedor MP4)

codificados en AAC, con content_type reportado como audio/mp4 o audio/mpeg. Esto concuerda con el comportamiento estándar de iOS y Android, que graban los clips de voz en este mismo formato.

Formato de las notas de voz

- **Contenedor:** MP4 (M4A)
- **Códec:** AAC
- **MIME:** audio/mp4 o audio/mpeg
- **Extensión:** .m4a

Este es el formato característico de los mensajes de voz en Messenger, mientras que otros audios cargados como adjuntos pueden conservar formatos distintos (por ejemplo, MP3 o WAV).

Cómo se recupera el archivo

Cuando se recibe un mensaje con nota de voz, la API retorna un objeto message que contiene:

- attachments[].type → "audio"
- attachments[].payload.url → URL temporal del archivo

El procedimiento es:

1. Capturar el evento message mediante el webhook de Messenger.
2. Extraer la URL en attachments[].payload.url.
3. Realizar una petición GET a esa URL con un Page Access Token válido.
4. Descargar el archivo resultante en formato M4A/AAC.

Implicaciones para STT

El formato nativo de las notas de voz en Messenger es M4A con códec AAC, ampliamente soportado por motores de Speech-to-Text modernos. En la mayoría de los casos puede utilizarse directamente. Sin embargo, en entornos que requieran compatibilidad universal, se podría convertir previamente a WAV PCM mono antes de la transcripción.

Resumen Comparativo de Formatos de Audio por Plataforma

Plataforma	Contenedor	Códec	Canal	MIME Type	Tamaño máx.	Conversión sugerida para STT
Google Chat	M4A	AAC	Mono	audio/mpeg	-	WAV PCM mono
WhatsApp	OGG	Opus	Mono	audio/ogg	16 MB	WAV PCM mono
Teams	MP4	AAC	Mono/Stereo	audio/mp4	-	WAV PCM mono
Telegram	OGG	Opus	Mono	audio/ogg	-	WAV PCM mono
Messenger	M4A	AAC	Mono	audio/mp4	-	WAV PCM mono

Motores de Transcripción de Voz Recomendados (STT – Speech-to-Text)

En el contexto del proyecto, el componente de transcripción automática de voz (STT) es esencial para transformar los audios recibidos desde múltiples plataformas (Google Chat, WhatsApp, Teams, Telegram y Messenger) en texto legible.

La elección del motor adecuado debe basarse en precisión, soporte de formatos, facilidad de integración, costo y disponibilidad regional.

A continuación, se presentan las principales opciones disponibles en la nube, junto con una alternativa open-source recomendada para entornos controlados.

Azure Speech Service (Microsoft)

El servicio Azure Speech to Text destaca por su alta precisión y compatibilidad con diferentes códecs, incluidos AAC, Opus, MP3 y WAV, lo cual facilita la integración directa con los formatos analizados en el documento.

Ofrece SDKs nativos para C#, Python, JavaScript y Java, ideal para una arquitectura basada en .NET como la que se plantea en este proyecto.

Ventajas:

- Procesamiento en tiempo real y transcripción por lotes.
- Reconocimiento automático de idioma (Auto Language Detection).
- Soporte para personalización mediante Custom Speech Models.

- Integración nativa con Azure Cognitive Services, facilitando un pipeline unificado.

Limitaciones:

- Requiere una suscripción activa en Azure y la configuración de una región específica (por ejemplo, eastus2 o centralus).
- Costos variables según minutos procesados, aunque con opciones de escalado controlado.

Conclusión:

Por su compatibilidad con tecnologías Microsoft, facilidad de integración y bajo nivel de latencia, Azure Speech Service se considera la opción más adecuada para la implementación inicial del proyecto.

AWS Transcribe (Amazon Web Services)

AWS Transcribe es una solución robusta y madura, orientada a aplicaciones empresariales que requieren transcripción masiva o análisis posterior del texto (por ejemplo, búsqueda semántica o etiquetado de sentimientos).

Admite formatos como MP3, MP4, WAV y FLAC, y ofrece tanto transcripción en streaming como procesamiento asíncrono por lotes.

Ventajas:

- Alta precisión y estabilidad en transcripciones prolongadas.
- Detección automática de idioma y segmentación de hablantes (speaker diarization).
- Soporte para integración con otros servicios de AWS (S3, Comprehend, Lambda).
- Seguridad y cumplimiento normativo de nivel empresarial (HIPAA, ISO, etc.).

Limitaciones:

- Costos más altos en comparación con otras plataformas.
- Configuración más técnica para el manejo de credenciales y permisos IAM.

Conclusión:

Recomendado para escenarios de gran volumen de audio o donde se necesiten análisis complementarios sobre los textos transcritos.

Google Cloud Speech-to-Text

Google Cloud STT es uno de los motores más precisos del mercado, especialmente para audios comprimidos como OGG/Opus (nativo de WhatsApp y Telegram).
Su motor aprovecha modelos neuronales de última generación y permite la transcripción multilingüe, incluyendo español latinoamericano y variantes regionales.

Ventajas:

- Soporte directo para formatos OGG/Opus, AAC y FLAC sin necesidad de conversión previa.
- Modelos especializados (telefónico, vídeo, meeting, etc.).
- API REST simple y flexible con bibliotecas en múltiples lenguajes.
- Precisión superior en entornos con ruido o diferentes acentos.
-

Limitaciones:

- Requiere gestión de credenciales con Google Cloud IAM.
- Puede presentar una latencia ligeramente mayor frente a Azure en escenarios en vivo.

Conclusión:
Es ideal para proyectos que requieran **alta compatibilidad con audios móviles** y diversidad lingüística.
En combinación con tu pipeline de normalización, **GCP STT** garantizaría interoperabilidad y precisión.

Whisper (OpenAI)

Whisper es un modelo de código abierto desarrollado por OpenAI que ofrece resultados competitivos con los principales servicios comerciales.

Su mayor ventaja radica en la posibilidad de ejecutarse localmente, lo cual elimina dependencia de servicios externos y costos por uso.

Ventajas:

- Precisión sobresaliente incluso con ruido ambiental.
- Admite una amplia variedad de formatos de audio sin conversión.
- Permite personalización y despliegue local (ideal para entornos con políticas de privacidad).
- Gratuito y respaldado por una comunidad activa de desarrolladores.

Limitaciones:

- Requiere capacidad de cómputo moderada-alta (GPU recomendada).
- No ofrece transcripción en tiempo real, sino en procesamiento por bloques.

Conclusión:
Recomendado para **entornos experimentales o privados**, donde la confidencialidad de los datos sea prioritaria o se busque evitar costos recurrentes.

Tabla comparativa

Motor STT	Precisión	Tiempo real	Costo	Despliegue recomendado
Azure Speech Service	Alta	✅	Medio	Aplicaciones .NET, nube Microsoft
AWS Transcribe	Muy alta	✅	Alto	Volumen empresarial, pipelines complejos
Google Cloud STT	Muy alta	✅	Medio	Audios móviles, multilenguaje
Whisper (OpenAI)	Alta	❌ (batch)	Bajo / gratuito	Prototipos, entornos locales

Conclusiones en el Contexto de CRISP-ML

La fase de Ingeniería de Características en el contexto de este proyecto no se limita al tratamiento de variables numéricas o categóricas, sino que se centra en la preparación y estandarización de señales de audio obtenidas desde diferentes plataformas de mensajería.
Cada una de estas plataformas —Google Chat, WhatsApp, Teams, Telegram y Messenger— entrega los archivos en formatos, códecs y estructuras distintas, lo que implica un trabajo previo de homogenización antes de poder aplicar modelos de transcripción o análisis.

Siguiendo la metodología CRISP-ML, esta etapa cumple con el propósito de garantizar la calidad, coherencia y utilidad de los datos, asegurando que el flujo posterior (transcripción automática, almacenamiento o análisis textual) sea confiable y reproducible.
El pipeline de preparación diseñado considera procesos equivalentes a los usados en proyectos de aprendizaje automático tradicional, tales como:

Normalización: conversión de todos los audios a un formato estándar (WAV PCM mono, 16 kHz).

Reducción de ruido y silencios: mejora de la calidad de señal y reducción de variables irrelevantes.

Estandarización de metadatos: asignación de identificadores únicos y trazabilidad de origen.

Selección de formato óptimo: análisis comparativo de compatibilidad con motores STT.

De esta forma, el documento demuestra que la preparación de los datos no depende exclusivamente de un entorno tabular, sino que puede representarse mediante procesos de ingeniería y decisión técnica, los cuales cumplen el mismo rol dentro del ciclo de CRISP-ML: transformar los datos crudos del mundo real en insumos útiles y estructurados.

Asimismo, la comparación de motores Speech-to-Text (Azure, AWS, Google Cloud y Whisper) aporta una base sólida para la selección del modelo más adecuado según los criterios de costo, rendimiento, escalabilidad y privacidad.
Esta decisión es clave dentro del paso de “Selección de Técnicas de Modelado” en CRISP-ML, ya que define la infraestructura técnica que soportará el aprendizaje o inferencia de texto a partir de audio.

Finalmente, este enfoque metodológico —basado en documentación técnica, análisis comparativo y normalización de procesos— sustituye adecuadamente el uso de una libreta de Jupyter.
Mientras que en proyectos tradicionales la exploración de datos se visualiza mediante código y gráficos, en este caso se logra mediante la formalización del flujo de audio, la estandarización de entradas y la evaluación de compatibilidad tecnológica.
Con ello, se cumple plenamente el objetivo de CRISP-ML: obtener datos preparados, justificados y listos para alimentar modelos o servicios de Machine Learning, en un entorno reproducible, escalable y documentado.

Referencias

Google Chat

Google Developers – *REST Resource: spaces.messages*
https://developers.google.com/workspace/chat/api/reference/rest/v1/spaces.messages

Google Developers – *Message.attachment*
https://developers.google.com/workspace/chat/api/reference/rest/v1/spaces.messages#attachment

Google Developers – *Download media attachments*
https://developers.google.com/workspace/chat/download-media-attachments

WhatsApp

Meta. WhatsApp Cloud API: Audio Messages. https://developers.facebook.com/docs/whatsapp/cloud-api/messages/audio-messages

Meta. WhatsApp Cloud API Webhooks: Messages – Audio Object.
https://developers.facebook.com/docs/whatsapp/cloud-api/webhooks/reference/messages/audio

Meta. WhatsApp Cloud API: Media Reference. https://developers.facebook.com/docs/whatsapp/cloud-api/reference/media

StackOverflow. WhatsApp Cloud API webhook decrypt media type audio.
https://stackoverflow.com/questions/73141631/whatsapp-cloud-api-webhook-decrypt-media-type-audio

Microsoft Teams

Microsoft Support – *Record a video or audio clip in Microsoft Teams*.
https://support.microsoft.com/en-us/office/record-a-video-or-audio-clip-in-microsoft-teams-0c57dae5-2974-4214-9c46-7a2136386f1c

Microsoft Learn – *chatMessageAttachment resource type (Graph API)*.
https://learn.microsoft.com/en-us/graph/api/resources/chatmessageattachment?view=graph-rest-1.0

Microsoft Learn – *Get attachment (Graph API)*.
https://learn.microsoft.com/en-us/graph/api/attachment-get?view=graph-rest-1.0

Microsoft Learn – *Recording and transcription for Teams meetings*.
https://learn.microsoft.com/en-us/microsoftteams/teams-recording-policy

Microsoft Learn – *Get meeting transcripts and recordings using Graph APIs*.
https://learn.microsoft.com/en-us/microsoftteams/platform/graph-api/meeting-transcripts/overview-transcripts

Microsoft Learn – *Set up Cloud Voicemail*.
https://learn.microsoft.com/en-us/microsoftteams/set-up-phone-system-voicemail

Microsoft Support – File types supported for previewing files in OneDrive, SharePoint, and Teams
https://support.microsoft.com/en-us/office/file-types-supported-for-previewing-files-in-onedrive-sharepoint-and-teams-e054cd0f-8ef2-4ccb-937e-26e37419c5e4

Telegram

Telegram TDLib – inputMessageVoiceNote (clase):
https://core.telegram.org/tdlib/docs/classtd_1_1td__api_1_1input_message_voice_note.html

Telegram Bot API – voice (objeto de notas de voz):
https://core.telegram.org/bots/api#voice

Telegram Bot API – getFile (para obtener la ruta del archivo a partir de un file_id):
https://core.telegram.org/bots/api#getfile

Telegram Bot API – file (estructura con file_path):
https://core.telegram.org/bots/api#file

Ejemplo de descarga de archivos (Telegram Bot API):
https://core.telegram.org/bots/api#making-requests

Messenger

Messenger Platform – Messages (webhook events, audio attachments):
https://developers.facebook.com/docs/messenger-platform/reference/webhook-events/messages

Messenger Platform – Attachment Upload API (formatos soportados):
https://developers.facebook.com/docs/messenger-platform/reference/attachment-upload-api

CM.com – Messenger audio formats (incluye m4a):
https://developers.cm.com/messaging/docs/facebook-messenger