Interacción Persona-Máquina

U6: Interfaces basadas en reconocimiento de voz y sonido (II)

Javier Rodríguez Juan
j.rodriguezjuan@ua.es
José García Rodríguez
jgr@ua.es

Universidad de Alicante

2025-2026

Grado en Ingeniería en Inteligencia Artificial

Universitat d'Alacant
Universidad de Alicante

3DPERCEPTIONLAB

dtic

Contenidos

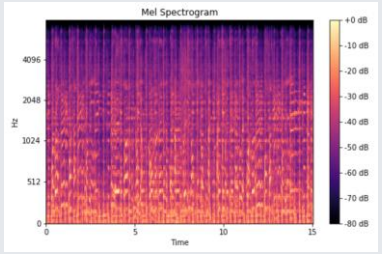
- Repaso
- Introducción al ASR
- El progreso
- El estado del arte
- ASR en la práctica

Repaso

Repaso: Preprocesamiento

- Al capturar un audio se deben realizar varios pasos de procesamiento antes de introducir este como entrada de la arquitectura de IA.

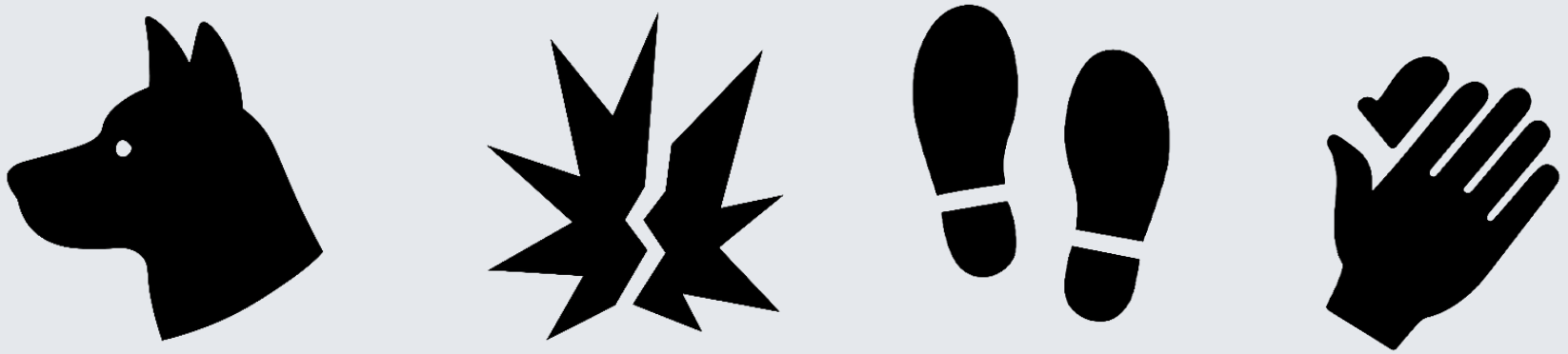
- Es común usar CNNs/Transformer encoders para generar representaciones más informativas. A estas las llamamos *embeddings*



$$features = [x1, x2, x3, \dots]$$

Repaso: ESR

Nuestros entornos están llenos de **sonidos** relevantes: timbres, pasos, aplausos, sirenas, ladridos, rotura de cristales, entre muchos más. El **reconocimiento de sonidos ambientales** (también llamado *Environmental Sound Recognition*, ESR) es la capacidad de un sistema para identificar y clasificar estos sonidos no verbales.



Repaso: ASR vs ESR

Similitudes	Diferencias
En ambos campos predomina el uso de redes neuronales como CNNs o Transformers	Mientras que el análisis del habla tiene una **estructura sintáctica clara**, los sonidos son más diversos.
	Escasez de datos para sonidos raros o eventos de corta duración.
	Mientras que dentro del ASR hay una **gran cantidad de subtareas (transcripción, subtitulado en streaming, clasificación)**, el ESR es más limitado y suele trabajarse como una tarea de clasificación de audio

Introducción al ASR

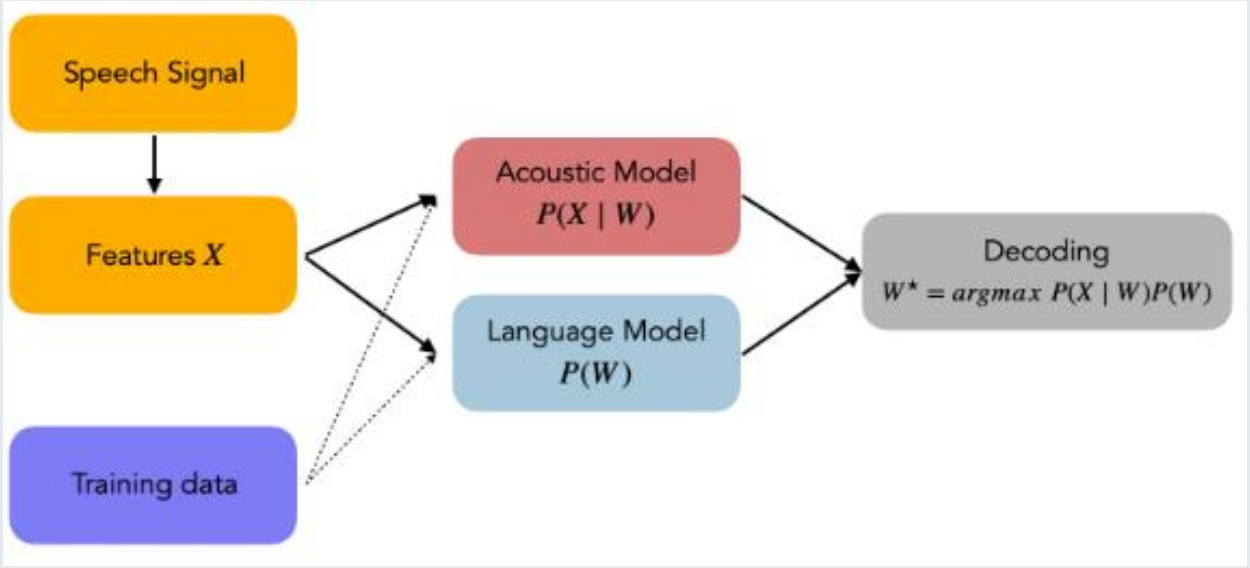
ASR: ¿Qué es?

El *Automatic Speech Recognition (ASR)* es la capacidad de una máquina para transformar el lenguaje natural en un formato interpretable para ella. Dentro de este se encuentra la tarea *del Speech-To-Text (STT)* donde a partir de un audio de entrada, se obtiene su transcripción asociada.

⚠️ No hay que confundir el ASR con el STT. El STT es la tarea de pasar de voz a texto mientras que el ASR abarca todas las técnicas involucradas con el análisis de audio, cómo el análisis de emociones, la diarización de locutores o incluso la predicción del deterioro cognitivo.

ASR: Los inicios

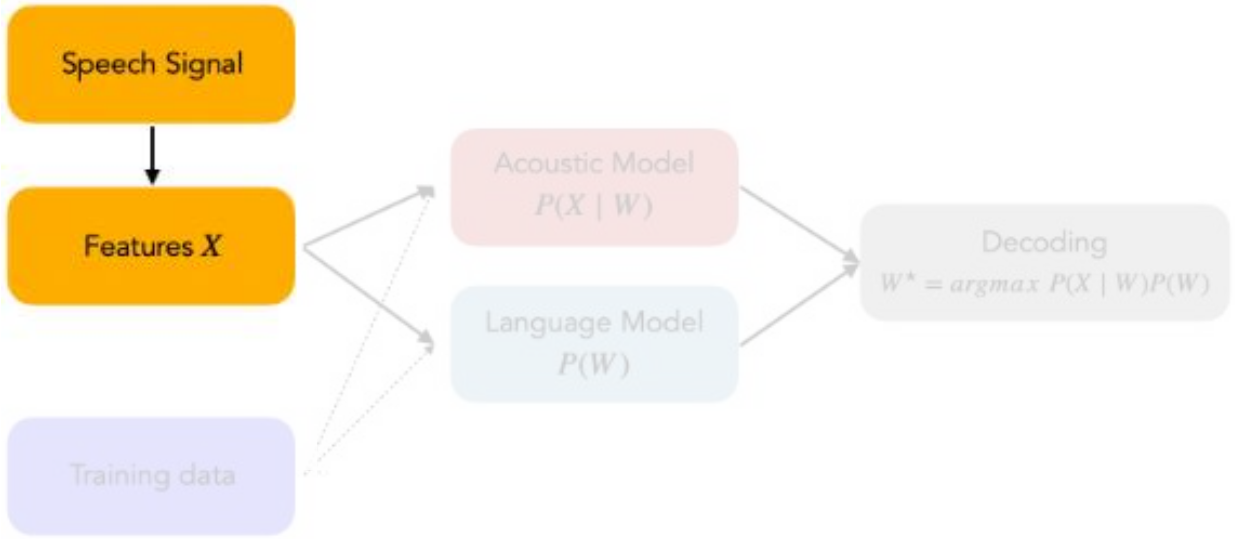
Inicialmente, los motores de ASR se componían de los siguientes módulos:



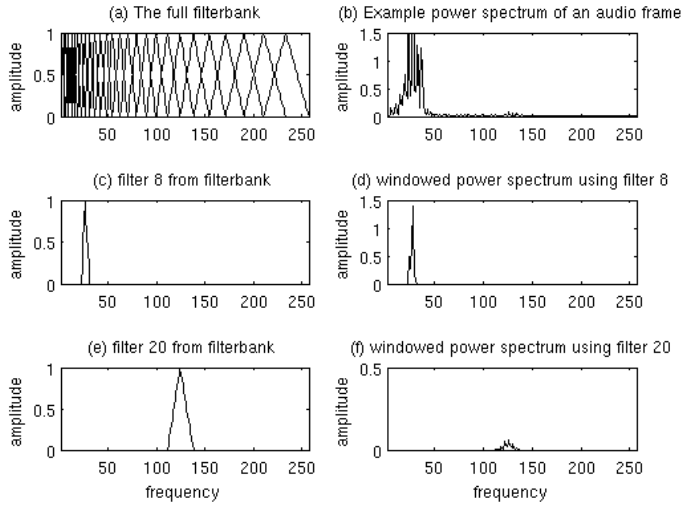
Componentes de un sistema de ASR. Extraído de [2].

ASR: Los inicios

1. **Captura de señal de audio:** Micrófono codifica señal
2. **Extracción de *features*:** A partir de la codificación se obtiene vector de *features* (MFCC, PLP)



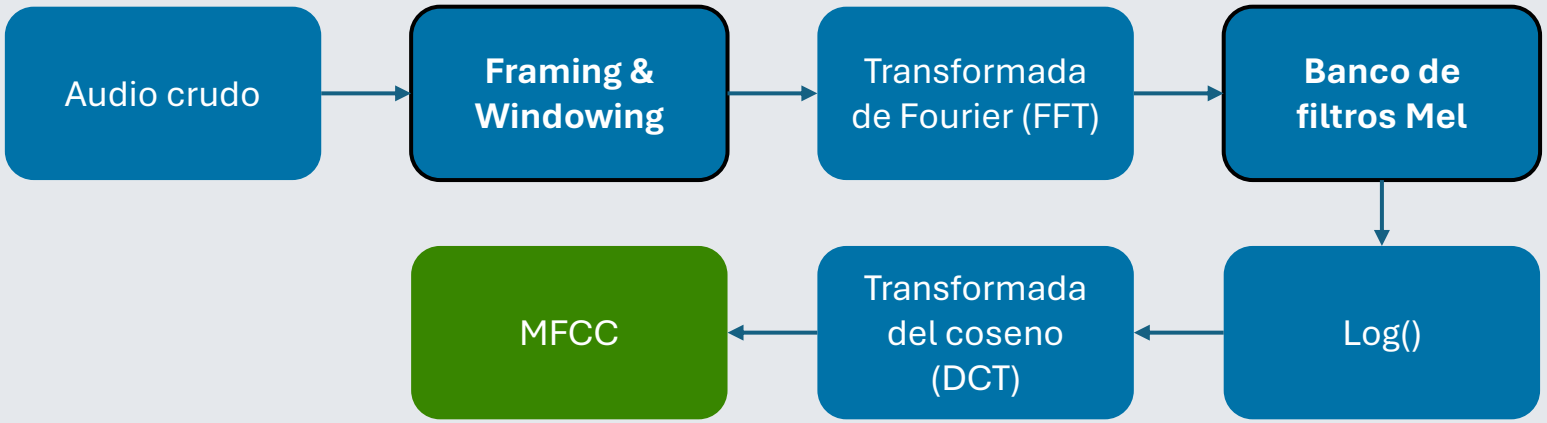
Componentes de un sistema de ASR. Extraído de [2].



Mel Filterbanks usados para calcular las features MFCC. Extraído de [2].

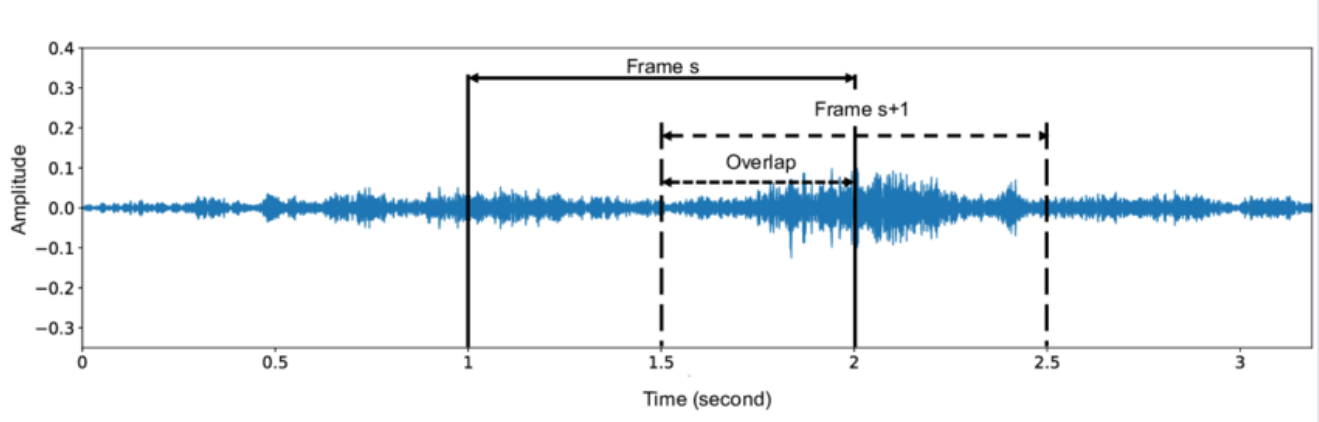
ASR: MFCC features

- Las features MFCC son una forma común de representar el audio. Su uso es muy común a la hora de realizar tareas de reconocimiento de audio, como la detección de enfermedades o la identificación de hablantes.
- Para extraer estas *features*, el proceso se compone de varios pasos. Por sencillez destacaremos los dos más importantes: el *windowing* y la aplicación de los filtros de Mel.



ASR: MFCC, Extracción

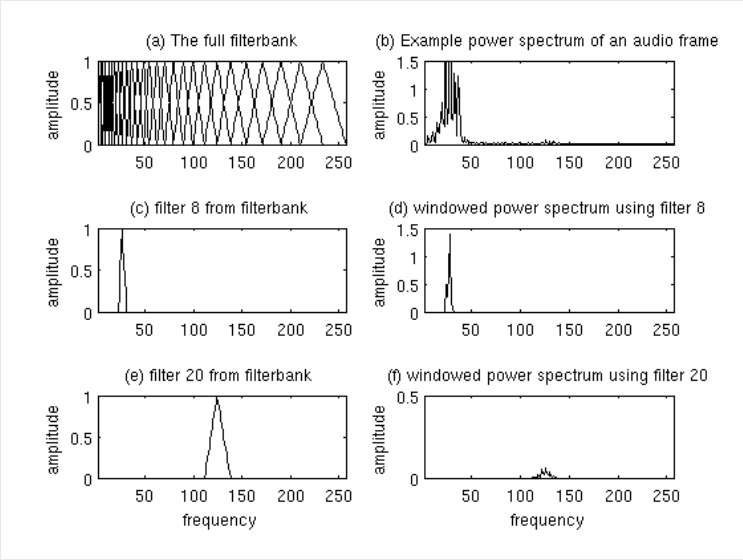
- **Windowing:** Un audio no se procesa como una unidad, sino que se separa en *frames*, mediante el proceso de *framing*. Después a cada *frame* se le aplica una ventana para reducir las discontinuidades en los bordes. Cada unidad del audio a analizar son las ventanas creadas tras el *windowing*.



Proceso de windowing. Extraído de [9]

ASR: MFCC, Extracción

- **Banco de filtros Mel:** Después del *windowing*, se le aplica a cada ventana la transformada de Fourier para obtener el espectro de magnitud del *frame*. Este espectro es el que se pasa por un conjunto de filtros triangulares distribuidos en la escala *Mel* (escala perceptual del oído humano). Cada filtro extrae la energía en una banda de frecuencias.



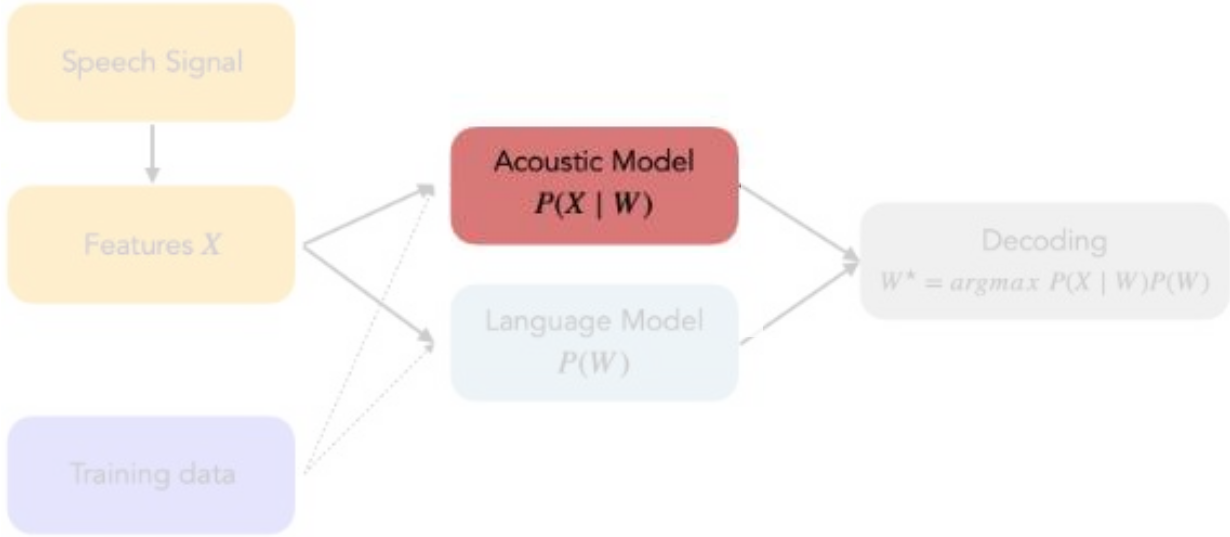
Mel Filterbanks usados para calcular las features MFCC.
Extraído de [2].

ASR: Los inicios

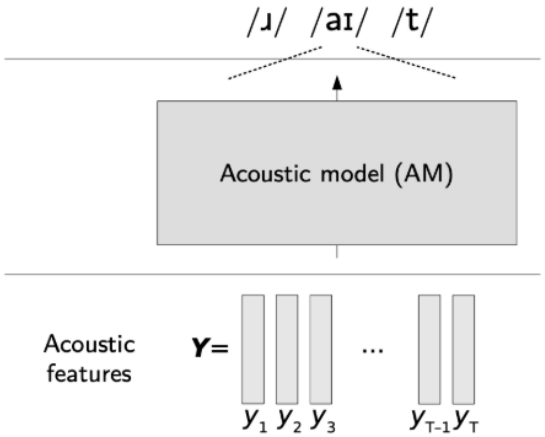
3. **Modelo acústico (AM):** Clasifica *features* en unidades fonéticas (HMM [1], ANN). Fonemas se convierten en palabras con diccionario fonético.

Ejemplo de *diccionario fonético:*

- "cat" → K AE T
- "cut" → K AH T
- "saw" → S AO



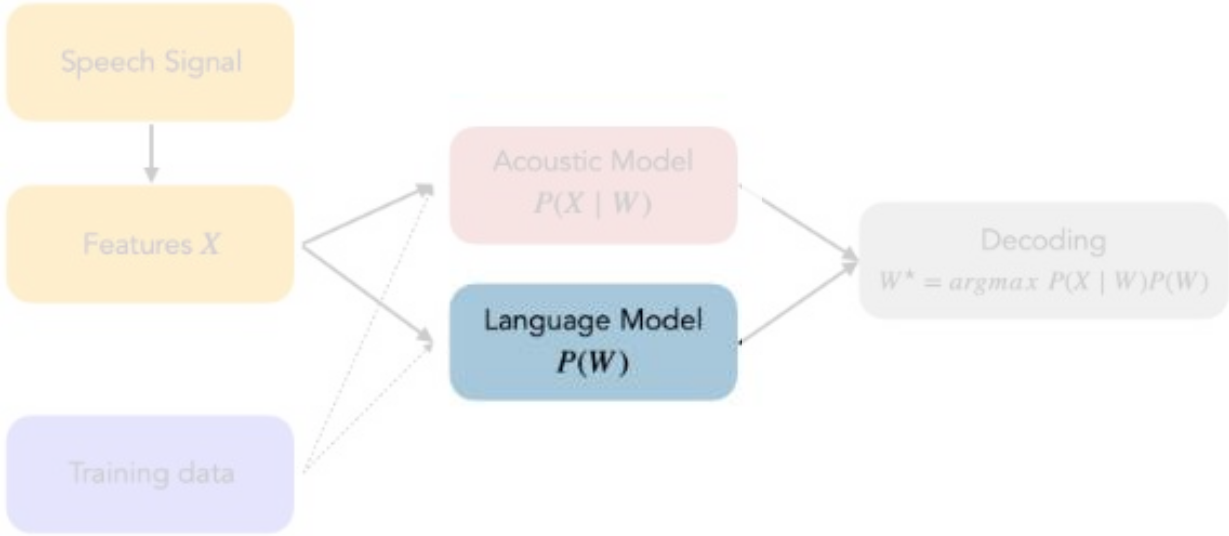
Componentes de un sistema de ASR. Extraído de [2].



Representación de un modelo acústico. Extraído de [2].

ASR:
Los inicios

4. **Modelo de lenguaje (LM):** A partir de secuencias textuales del AM el LM favorece opción más probable según contexto. Una práctica común es el uso de un *n-gram model*.



Componentes de un sistema de ASR. Extraído de [2].

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

Here are the calculations for some of the bigram probabilities from this corpus

$P(\text{I}|\text{<s>}) = \frac{2}{3} = 0.67$ $P(\text{Sam}|\text{<s>}) = \frac{1}{3} = 0.33$ $P(\text{am}|\text{I}) = \frac{2}{3} = 0.67$

$P(\text{</s>}|\text{Sam}) = \frac{1}{2} = 0.5$ $P(\text{Sam}|\text{am}) = \frac{1}{2} = 0.5$ $P(\text{do}|\text{I}) = \frac{1}{3} = 0.33$

Ejemplos de probabilidades de un modelo de bigramas (2-gram model). Extraído de [3].

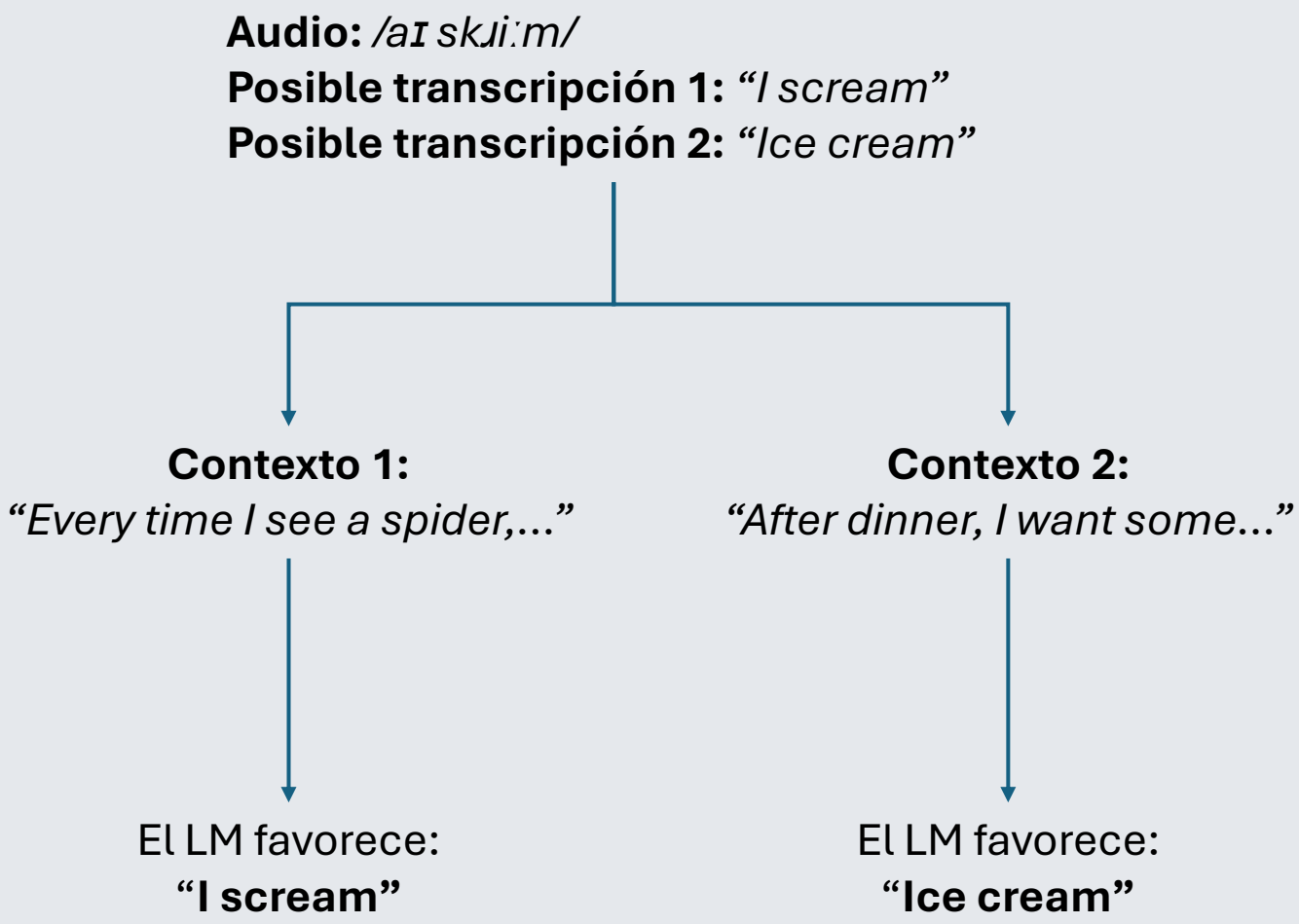
ASR: ¿Es suficiente el modelo acústico?

- NO! El modelo acústico puede confundir sonidos similares.
- El modelo de lenguaje (LM) usa el contexto para elegir la palabra más probable.

<u>Ejemplo:</u>

Fonema → /aɪ skɹiːm/

¿Cuál es su transcripción?

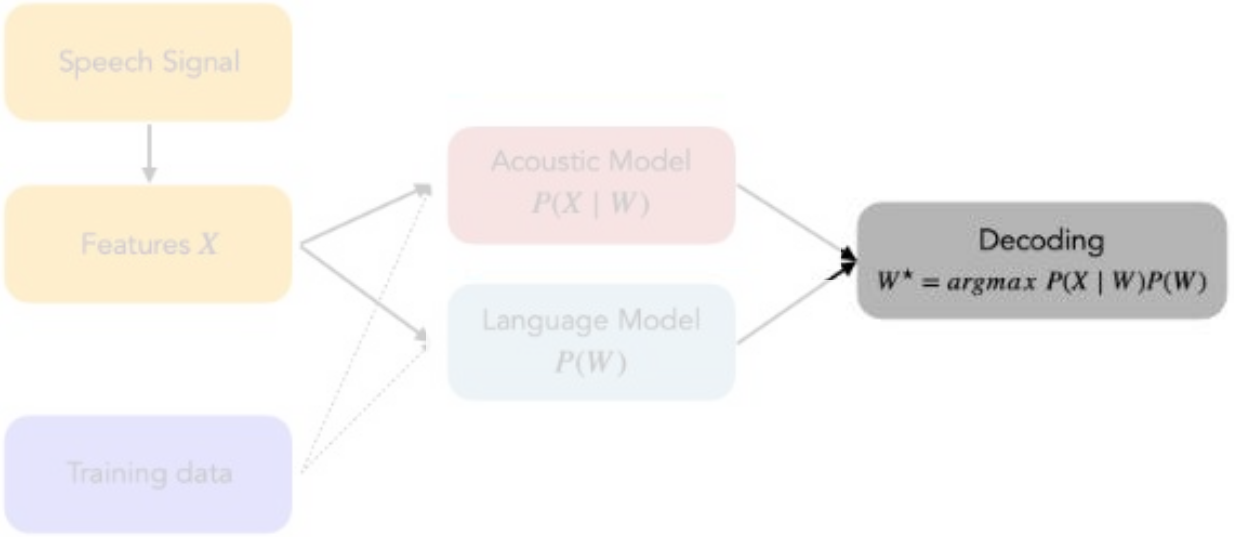


ASR: Los inicios

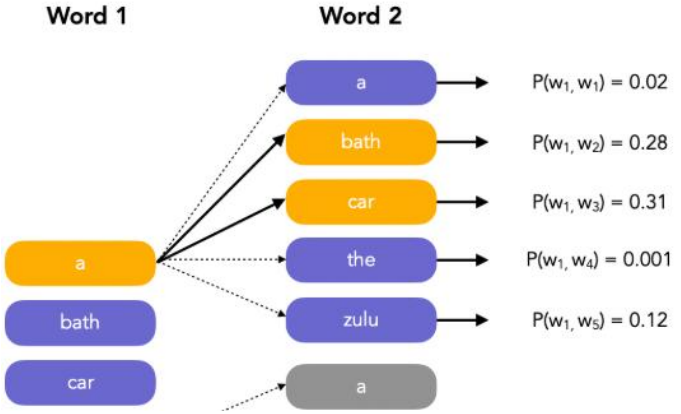
5. **Decodificación:** *Beam Search* [4] sobre el vocabulario de nuestro modelo maximizando la probabilidad:

$$argmax\ P(X|W) * P(W)$$

$$P(X|W) = \ prob.del\ AM$$
$$P(W) = prob.del\ LM$$



Componentes de un sistema de ASR. Extraído de [2].



Ejemplo de *beam search*. Las cajas amarillas son las palabras escogidas para continuar la búsqueda. Extraído de [2].

El progreso del ASR

El progreso: HMM-DNN

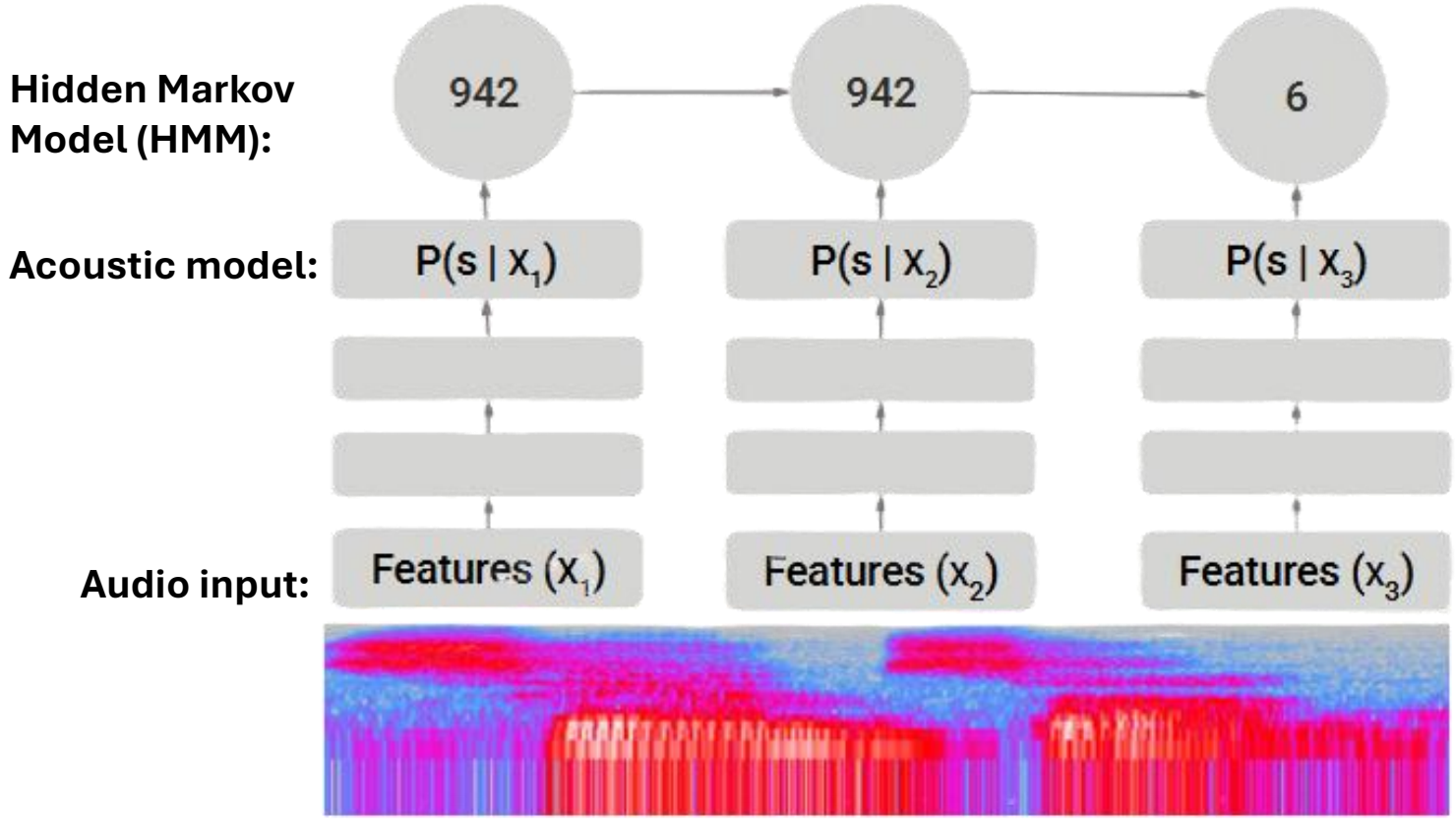
Se avanza hacia el uso de redes neuronales. La primera aproximación que incluyeron estas fueron los modelos HMM-DNN [5], dónde el modelo acústico es una *deep neural network* en vez de una GMM (mezcla gaussiana).

Posteriormente se fueron introduciendo en los modelos de ASR las RNNs, LSTMs y GRUs.

Pronunciation: Samson

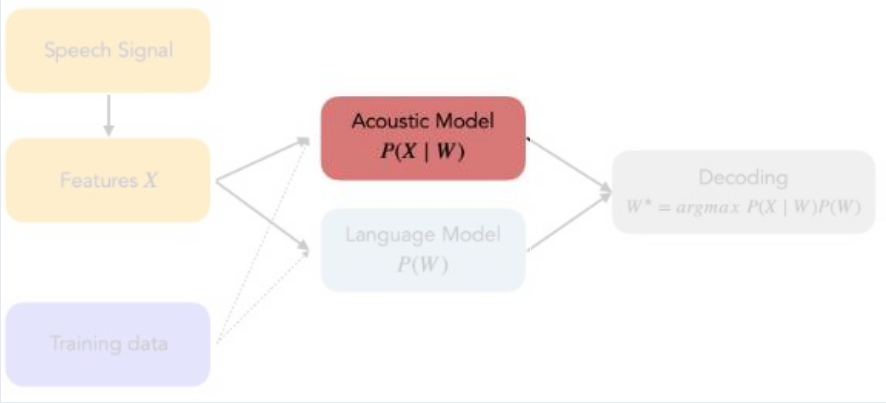
Pronunciation: S – AE – M – S – AH – N

Subphones: 942 – 6 – 37 – 8006 – 4422 - ...



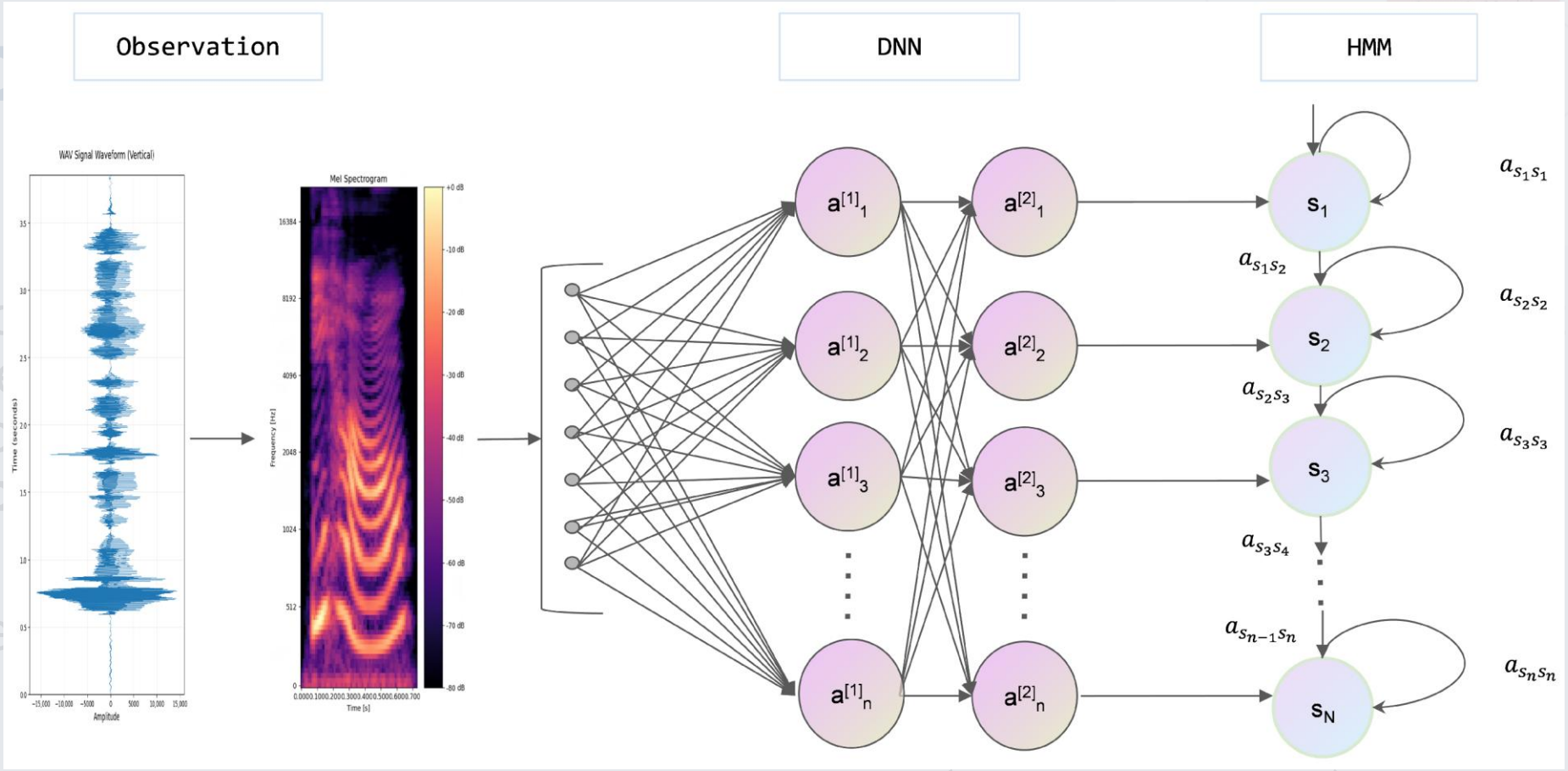
Arquitectura HMM-DNN. Extraído de [5].

El progreso: HMM-DNN



- La idea es introducir como entrada al modelo acústico (DNN) un vector de features MFCC, y este genera como salida no fonemas, sino probabilidades posteriores de estados HMM. Es decir, la DNN estima la **probabilidad de que un frame corresponda a cierto estado fonético**.
- El HMM define como de probable es pasar de un estado a otro. Cada fonema tiene tres estados: inicio, medio, final.

Lo que predice la DNN es la parte del fonema (inicio, medio, fin) que se está pronunciando.



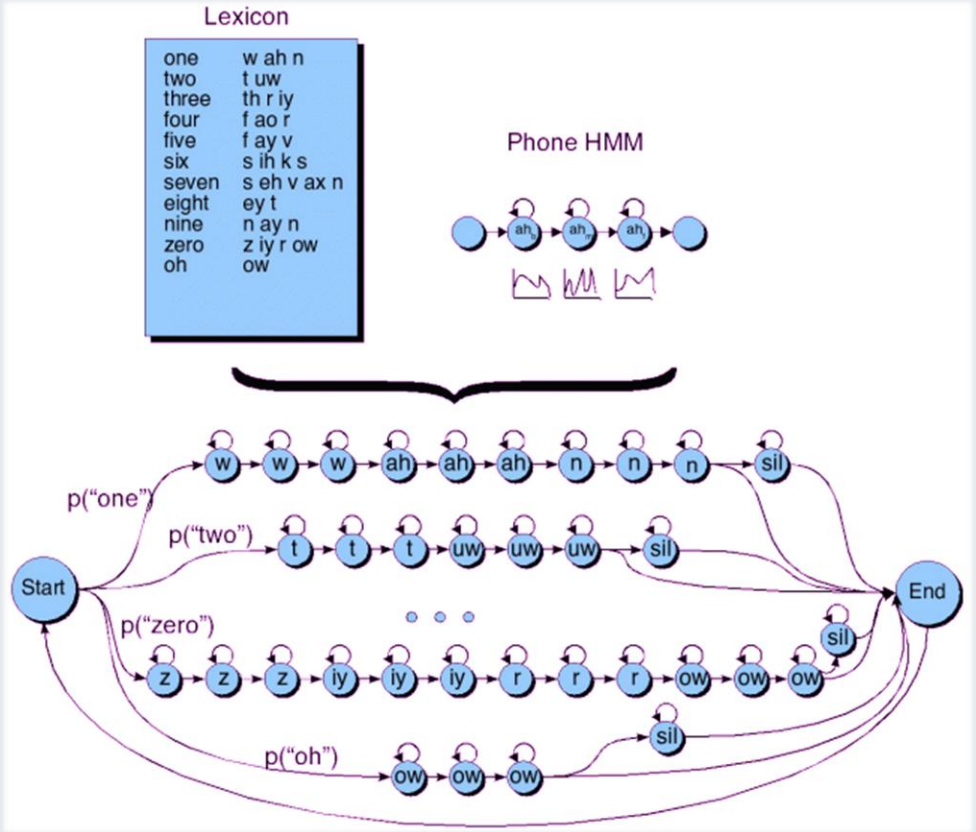
Estructura interna de la DNN y del HMM que componen una arquitectura HMM-DNN. Extraído de *Wu et al. (2025). Integrating Speech Recognition into Intelligent Information Systems: From Statistical Models to Deep Learning. Informatics.*

El progreso: HMM-DNN

ℹ️ ¿Qué es un HMM?

Modelo probabilístico que representa procesos que evolucionan en el tiempo con estados ocultos y observaciones visibles.

En ASR, modela el orden de los sonidos, decide cuanto dura cada fonema, nos ayuda a alinear frames con fonemas, y permite buscar la secuencia más probable de fonemas que corresponde al audio.



Ejemplo de HMM para reconocimiento de dígitos. Extraído de [5].

El progreso: HMM-DNN

Limitaciones

1. **Dependencia de un léxico diseñado manualmente.** Requiere ingeniería lingüística, lo que introduce suposiciones rígidas sobre la pronunciación.
2. **Pipeline de entrenamiento en múltiples etapas.** Primero se debe comenzar con un modelo HMM-GMM para obtener alineaciones iniciales, y luego usar esas alineaciones para entrenar el modelo acústico DNN.
3. **La modularidad limita la optimalidad.** Los sistemas HMM-DNN dividen el pipeline en distintos módulos que se entrenan por separado, lo que conduce a una solución globalmente subóptima.

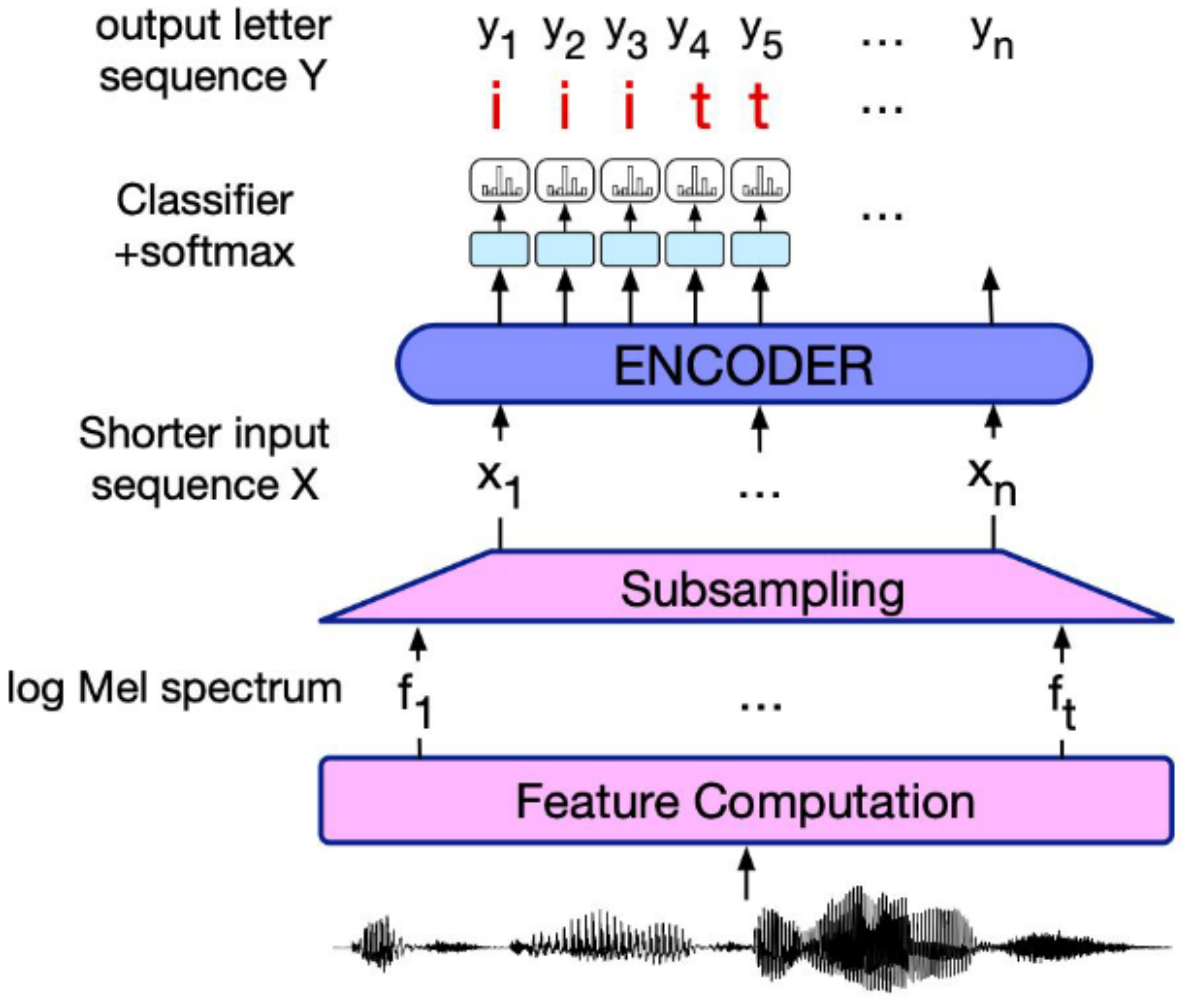
¿Puede el Deep Learning reemplazar completamente el uso de HMMs?

El progreso: CTC

El ASR progresó hacia las arquitecturas CTC [5], que predecían outputs textuales directamente de las entradas de audio.

Estos modelos predicen un carácter para cada timestamp *t*. Cada *t* es independiente del resto, una predicción no depende de la anterior.

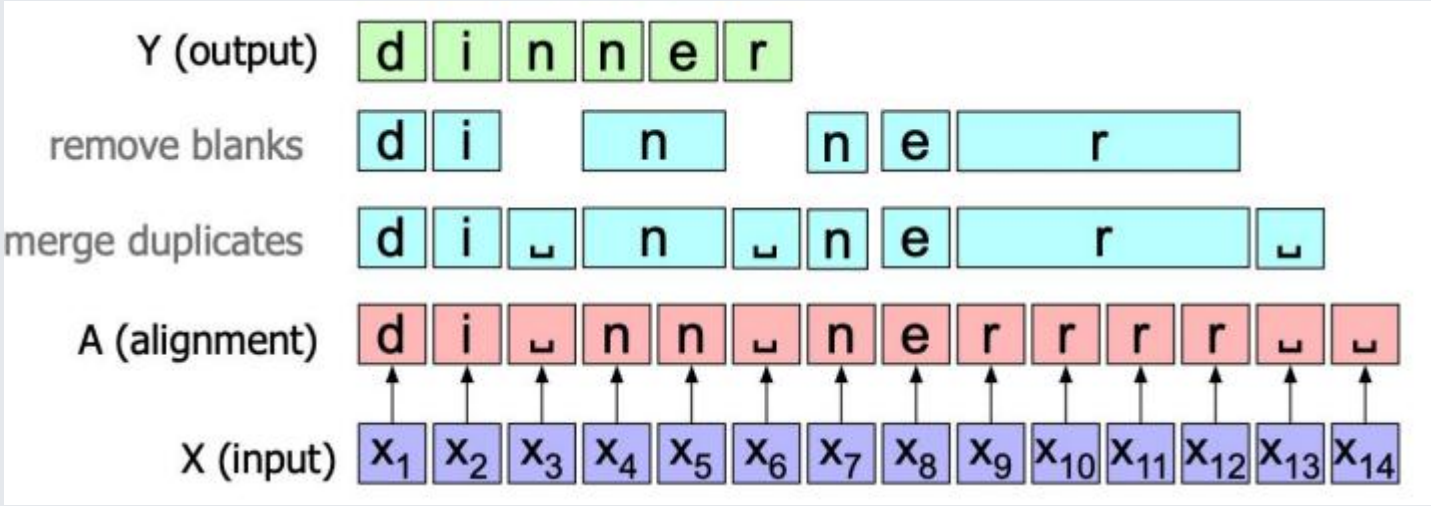
Aquí ya se puede observar la entrada de los mecanismos de Atención y de la arquitectura Transformer [3].



Arquitectura CTC. El módulo ENCODER es un Transformer Encoder. Extraído de [5].

El progreso: CTC

- Primero, en cada timestamp *t* el modelo produce un token de salida que puede ser un carácter o un “blank” (vacío)
- Después se aplica la regla de colapsamiento para generar la salida → Ignorar todos los “blanks”, e ignorar todas las salidas repetidas del mismo token



Regla del colapsamiento en CTC. Extraído de [3].

El progreso: CTC

Limitaciones

- **Asunción de independencia.** Asumir que la salida *t* es completamente independiente de la *t-1* limita la precisión de estos sistemas, ya que no se apoyan del contexto predicho hasta el momento para refinar las predicciones.

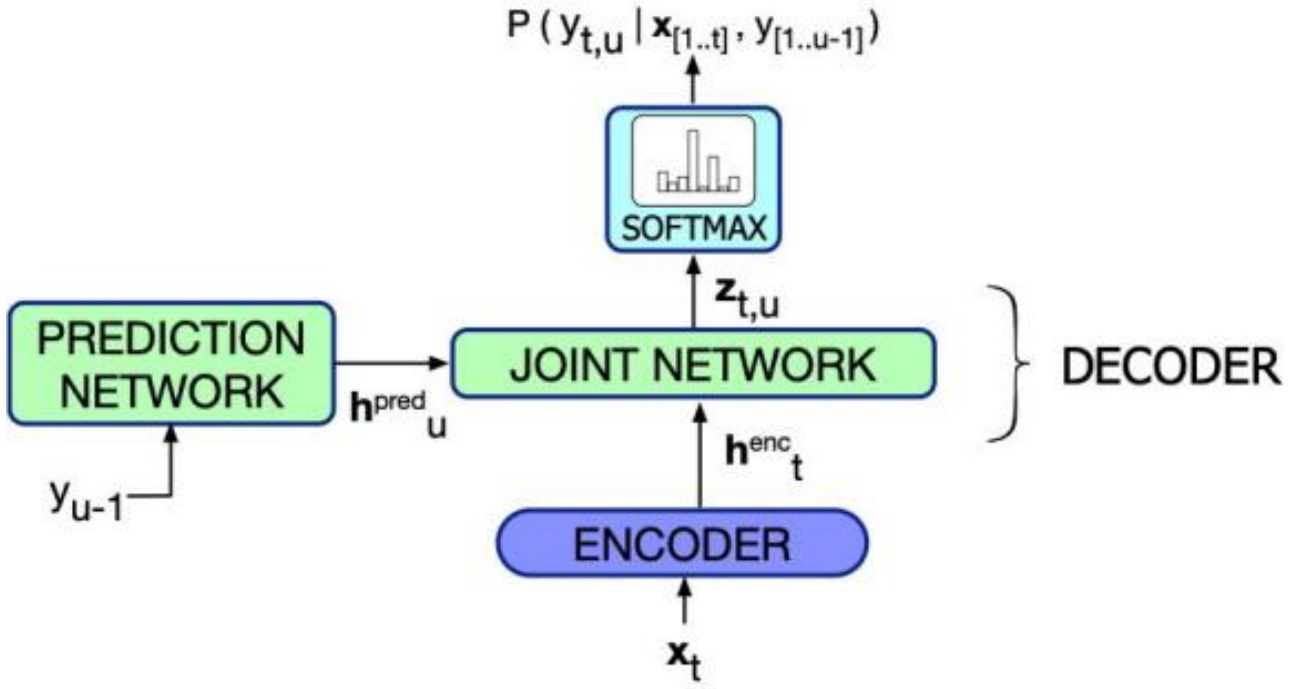
Ventajas

- **Streaming.** La asunción de independencia lo hace muy conveniente para usarse en *streaming*. *Streaming* significa reconocer las palabras online, en vez de esperar a que acabe toda la secuencia. Esto es crucial en aplicaciones donde queremos empezar el reconocimiento mientras que el usuario está hablando.

El progreso: RNN-Transducer

Para eliminar la asunción de independencia de las CTC, se introdujo el RNN-Transducer [8]. Este se compone de dos partes: un modelo acústico CTC y un modelo de lengua separado llamado *predictor*.

En cada timestamp *t*, el CTC produce una representación h_t dada la entrada $x_1 \dots x_n$. Después el *predictor* produce otro h_u a partir de los tokens de salida anteriores (contexto). Finalmente, h_t y h_u se introducen en una tercera red a cuya salida se le aplica una *softmax* para predecir el siguiente carácter.



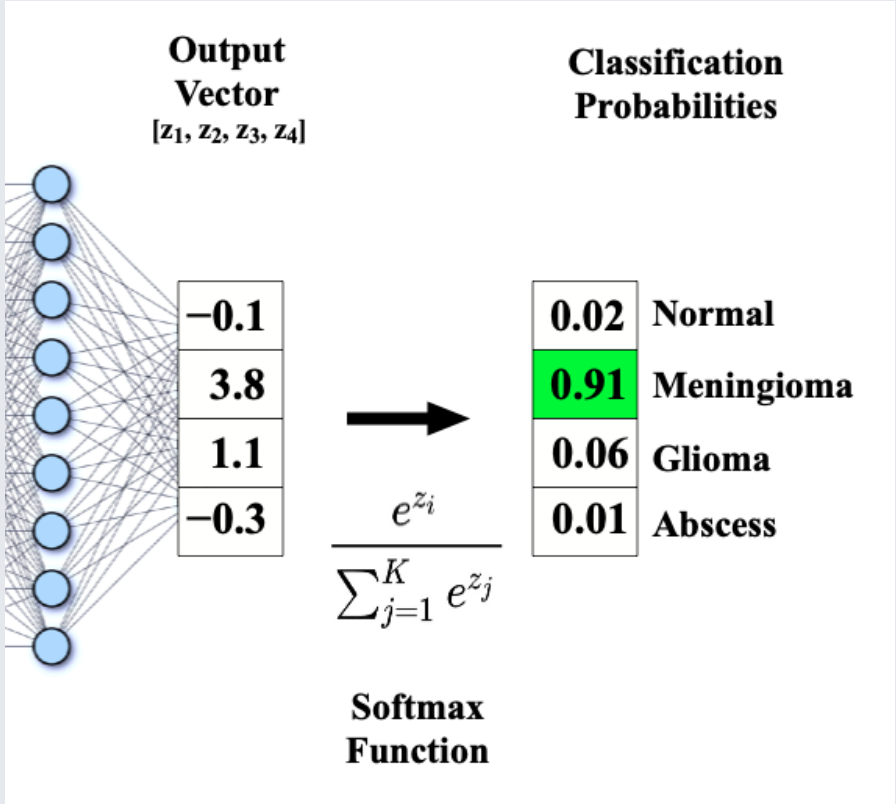
Arquitectura del RNN-Transducer. Extraído de [3].

El progreso: RNN-Transducer

ℹ️ ¿Qué es una *softmax*?

La capa *softmax* normaliza los valores de salida obtenidos por una red neuronal (comúnmente llamado *logits*) dentro del rango [0,1].

Esto convierte los *logits* en probabilidades por clase fácilmente interpretables.

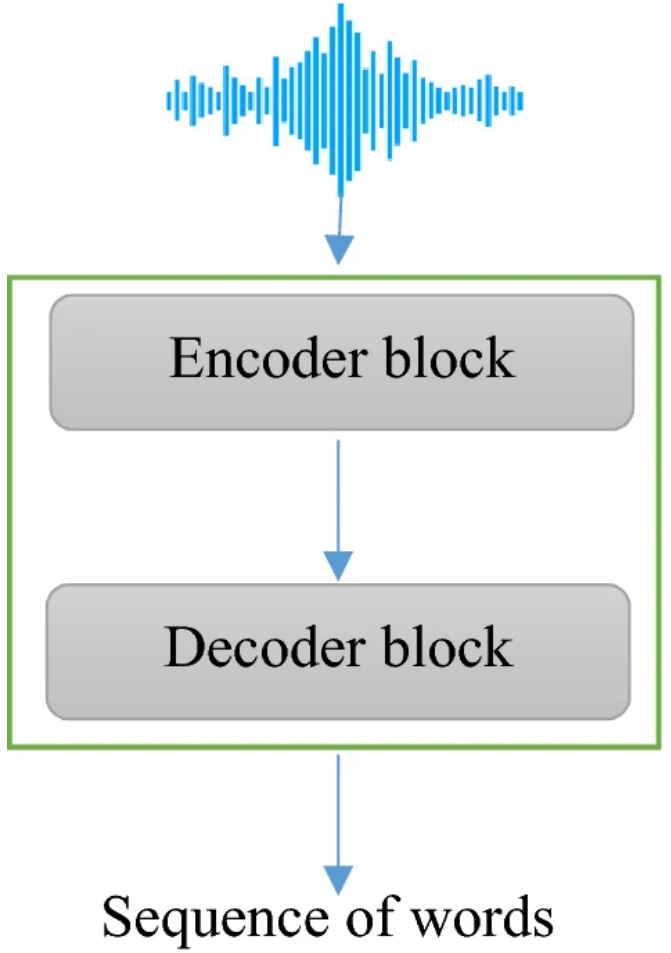


Ejemplo de clasificación con softmax. Extraído de https://mriquestions.com/softmax.html.

El progreso: Sequence-to-sequence

Y llegamos hasta los modelos *sequence-to-sequence (S2S),* donde el modelo es capaz de predecir secuencias enteras a partir del audio de entrada.

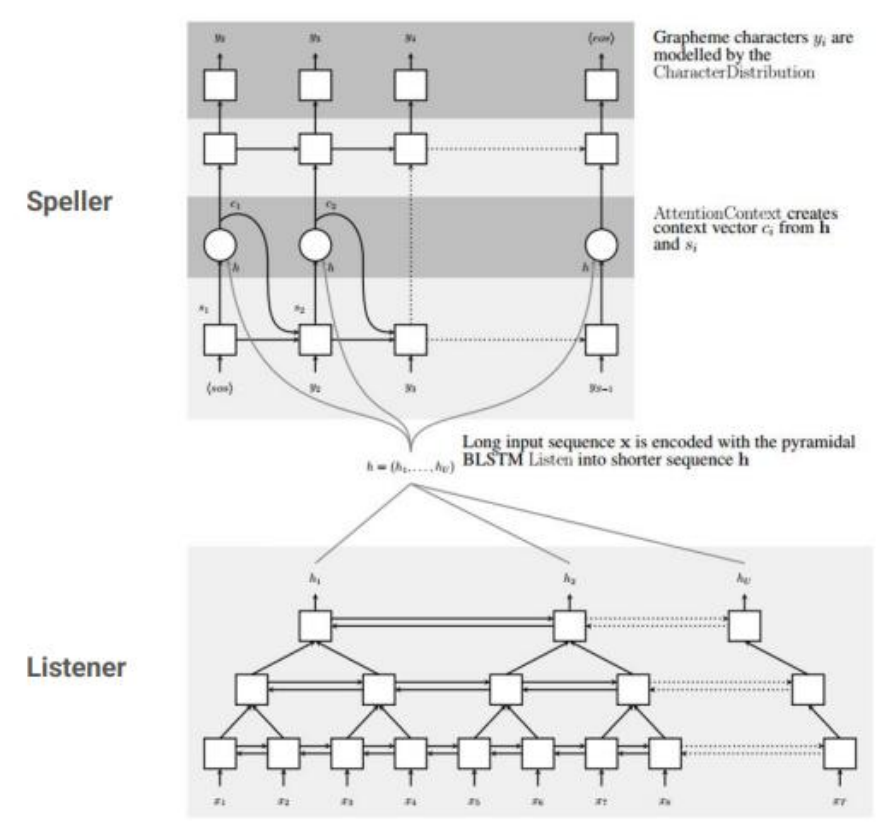
Los mejores resultados se logran con modelos Transformer o modelos híbridos CNN/Transformer. Con estos se consiguió sobre el 2016-2017 llegar a un nivel de transcripción de “paridad con humanos” con tasas de error de palabra ~5%.



Arquitectura genérica de un modelo S2S para ASR. Extraído de [6]

El progreso: Sequence-to-sequence

- El decoder suele incluir mecanismos de atención. Esto marcó el inicio de las arquitecturas encoder-decoder modernas.
- A diferencia de CTC, las S2S solo producen caracteres (no “blanks”).
- Una salida *t*, está condicionada por las salidas anteriores, es decir, por su contexto.



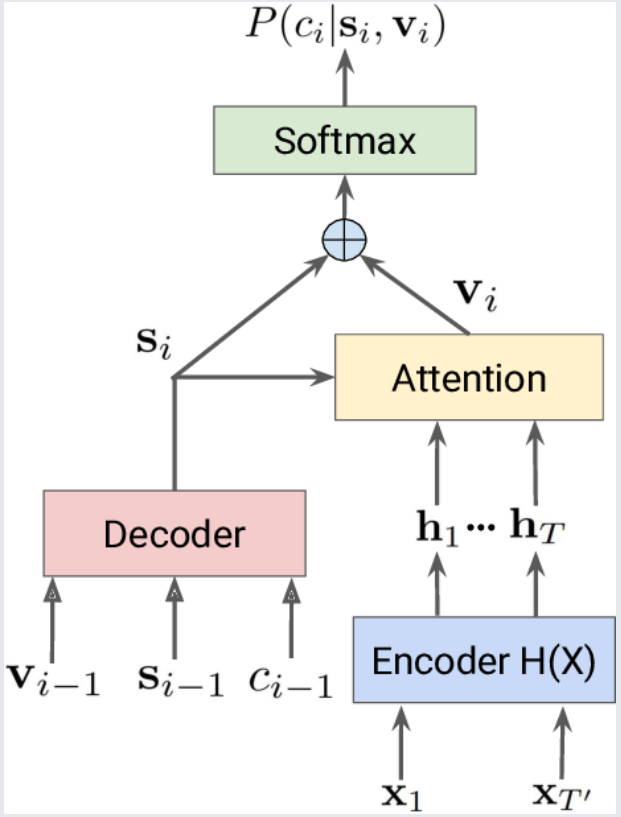
Arquitectura interna de un modelo S2S. En ASR, al encoder se le puede llamar Listener y al decoder Speller. Extraído de [6]

El progreso: Sequence-to-sequence

ℹ️ ¿Qué es la atención?

El mecanismo de atención permite que el modelo, en vez de mirar todo el audio a la vez, se enfoque solo en las partes más relevantes en cada momento.

Dentro de una arquitectura *encoder-decoder* para ASR, el encoder transforma el audio en una serie de representaciones, y el decoder las va utilizando para generar el texto. La atención actúa como una "linterna" que, en cada paso, ilumina la parte del audio que es más útil para predecir la siguiente letra o palabra.



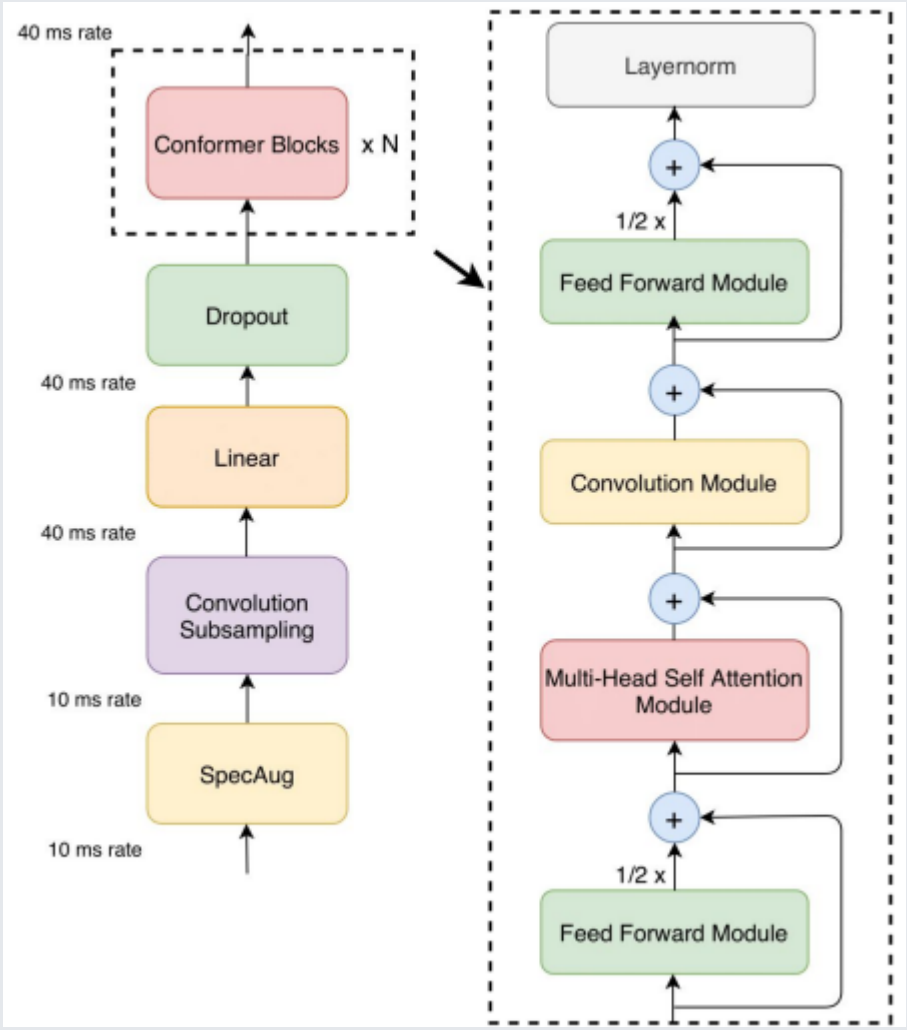
Arquitectura encoder-decoder con uso de atención. Extraído de Prabhavalkar et al. (2023). *End-to-end speech recognition: A survey*

Estado del arte del ASR

Estado del arte: Conformer (Google)

La arquitectura **Conformer** (Convolution-augmented Transformer) nació para mejorar el rendimiento en ASR combinando lo mejor de dos mundos:

- La capacidad del **Transformer** para capturar dependencias globales en secuencias largas
- La fortaleza de las **convoluciones** para modelar patrones locales y variaciones temporales del habla.

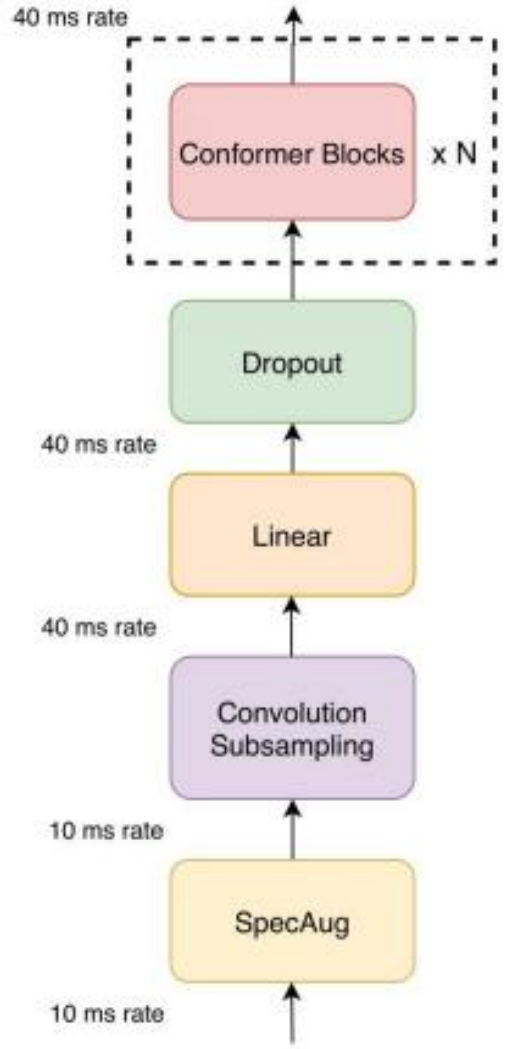


Arquitectura del modelo Conformer. Extraído de [5]

Estado del arte: Conformer, Encoder

Se compone de diferentes partes:

- **SpecAug:** técnica de *data augmentation* aplicada directamente a los espectrogramas del audio. En lugar de modificar la señal de audio original, se distorsionan ciertas partes del espectrograma.
- **ConvSubsampling:** Es una forma de *reducir la longitud de la secuencia de entrada*. En vez de procesar cada frame de audio uno por uno (lo cual es costoso), se aplican convoluciones con stride (saltos).

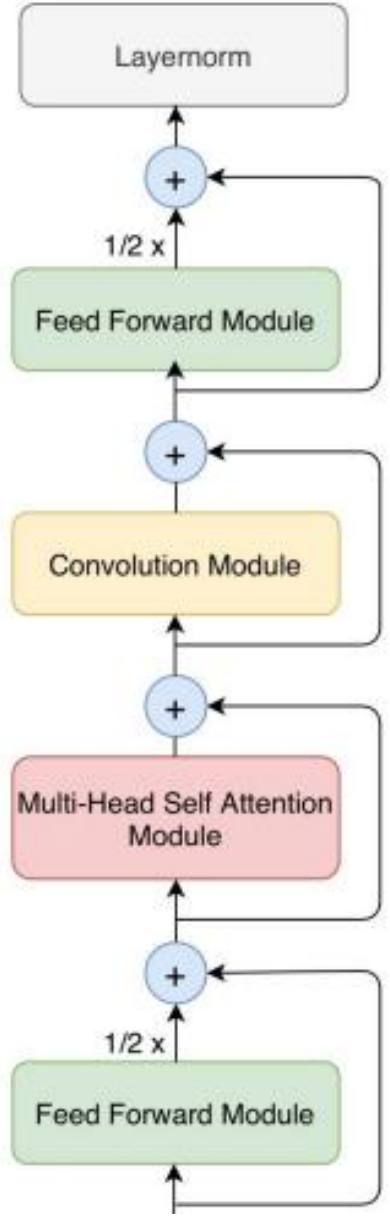


Arquitectura general del modelo Conformer. Extraído de [5]

Estado del arte: Conformer, Encoder

Se compone de diferentes partes:

- **Bloque Conformer (Conformer encoder):** Evolución del bloque Transformer diseñado específicamente para audio. Añade un bloque convolucional para capturar contexto local e introduce dos redes feedforward en lugar de una.

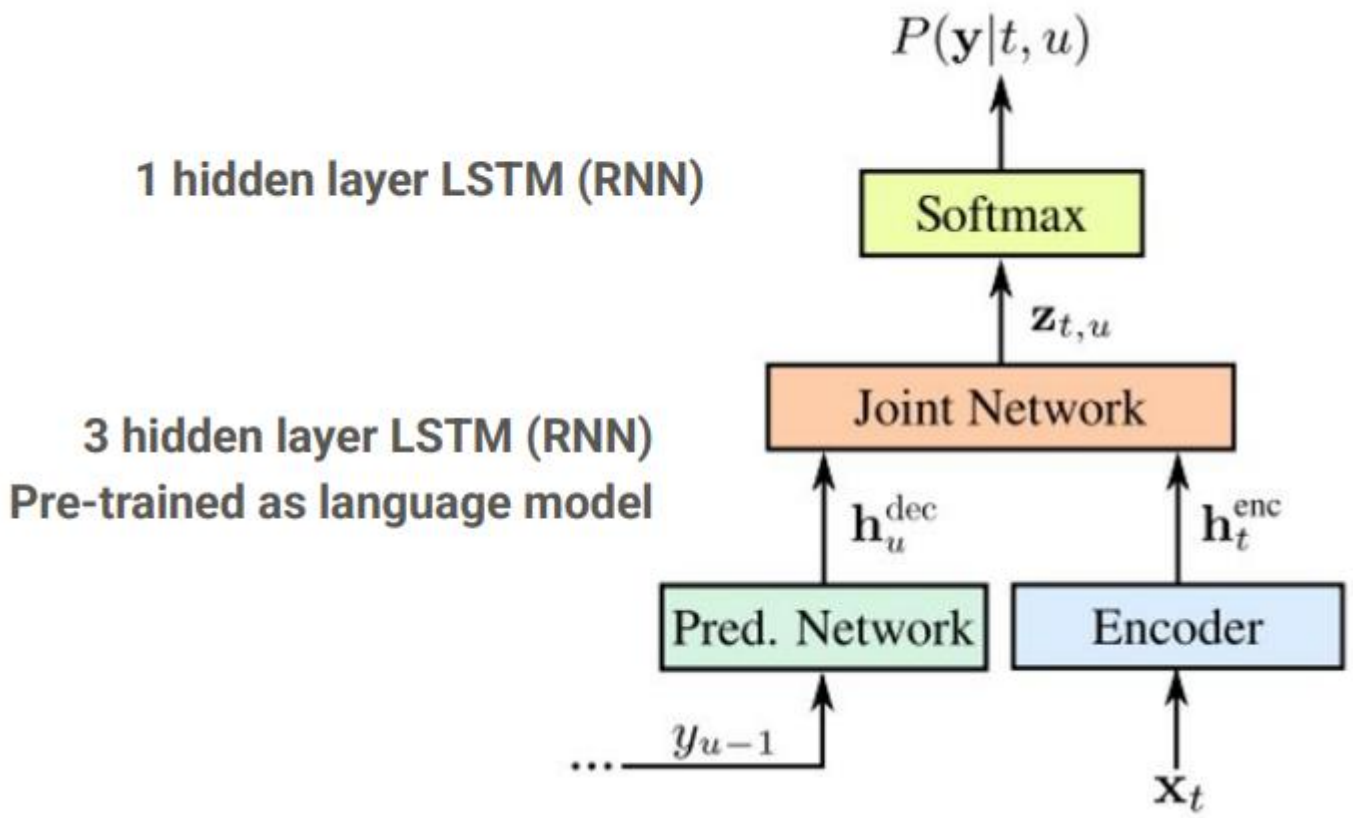


Arquitectura del bloque Conformer. Extraído de [5]

Estado del arte: Conformer, Decoder

¡Conformer no incluye ningún bloque decoder en su publicación original! El Conformer es una arquitectura que solo abarca la parte del encoder.

Para poder completar pipelines d ASR usando el Conformer, es común acompañar a este modelo de un Transformer decoder como el de Whisper o de una arquitectura RNN-T compuesta de redes recurrentes.



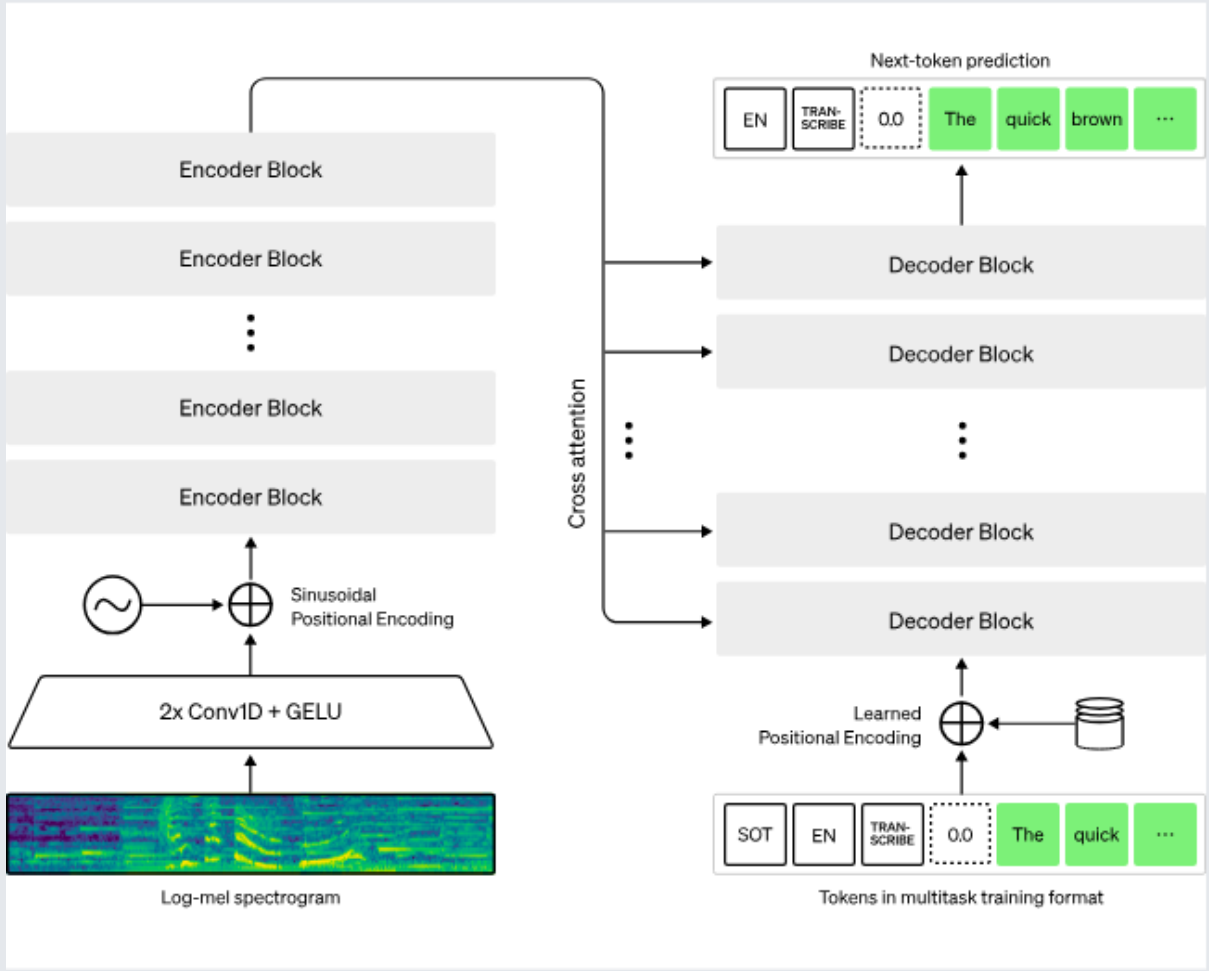
Posible decoder RNN-T para el modelo Conformer. Extraído de [5]

Estado del arte: Whisper (OpenAI)

Modelo *open-source* entrenado con 680.000 horas de audio multitarea y multilingüe. Es muy robusto a acentos, ruido de fondo y lenguaje técnico.

Además de transcribir en múltiples idiomas, puede también traducir del idioma hablado al inglés y generar marcas de tiempo por fragmento de frase.

Su arquitectura es la de un Transformer *encoder-decoder* estándar. Destaca el uso de una CNN para extraer *features* acústicas a partir de un espectrograma de Mel.



Arquitectura del modelo Whisper. Extraído de https://openai.com/index/whisper/

Estado del arte: Whisper

¿Qué puede hacer Whisper?

- **Transcripción inglés**

🗣️ “Ask not what your country…” → 📝 “Ask not what your country…”

- **Transcripción y traducción al inglés**

🗣️ “El rápido zorro marrón…” → 📝 “The quick brown fox…”

- **Transcripción de cualquier otro idioma**

🗣️ “Un caffè e una torta per favore…” → 📝 “Un caffè e una torta per favore…”

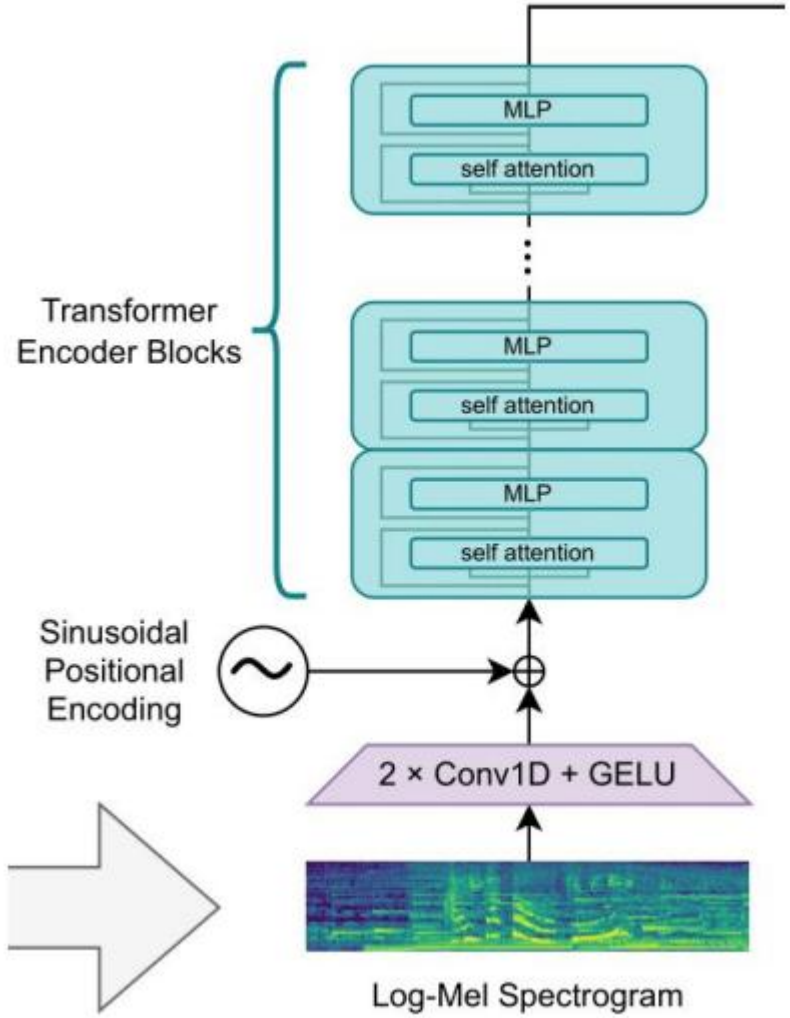
- **Identificar cuando no hay hablantes**

🗣️ (música de fondo)→ 📝 ∅

Estado del arte: Whisper, Encoder

El encoder se compone de tres partes principales

- **Entrada:** Es un espectrograma log-mel acompañado por dos capas convolucionales
- **Codificación posicional:** Es común que a la entrada de un Transformer se le aplique un embedding posicional para que cada parte del input mantenga información posicional.
- **Capa Transformer**

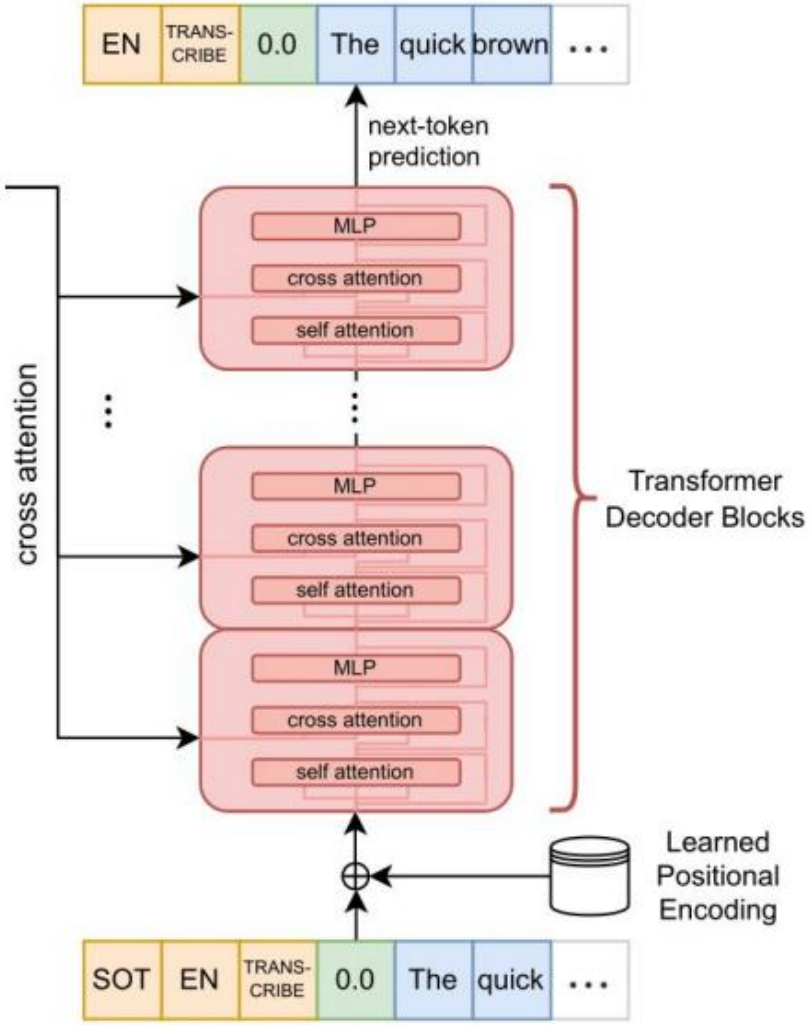


Arquitectura del Transformer Encoder de Whisper. Extraído de [5]

Estado del arte: Whisper, Decoder

En el decoder destaca lo siguiente

- **Estilo GPT-2:** El decoder es el mismo que se utiliza como decoder en la arquitectura GPT-2
- **Vocabulario/tags personalizados:** Para establecer la tarea a realizar Whisper añade tokens que “configuran” la tarea que se va a realizar en la inferencia (p. ej. Etiquetas de lenguaje como <en> o <fr>, etiquetas de tarea como <transcribe> o <translate>, etiquetas de habla como <nospeech>)



Arquitectura del Transformer Decoder de Whisper. Extraído de [5]

Estado del arte: Whisper

Ejemplo de salida

Ground truth: Comme moi, vous avez peut-être déjà vécu cette scène. Qu'est-ce que tu fais cet été toi? Écoute, je ne sais pas encore trop.

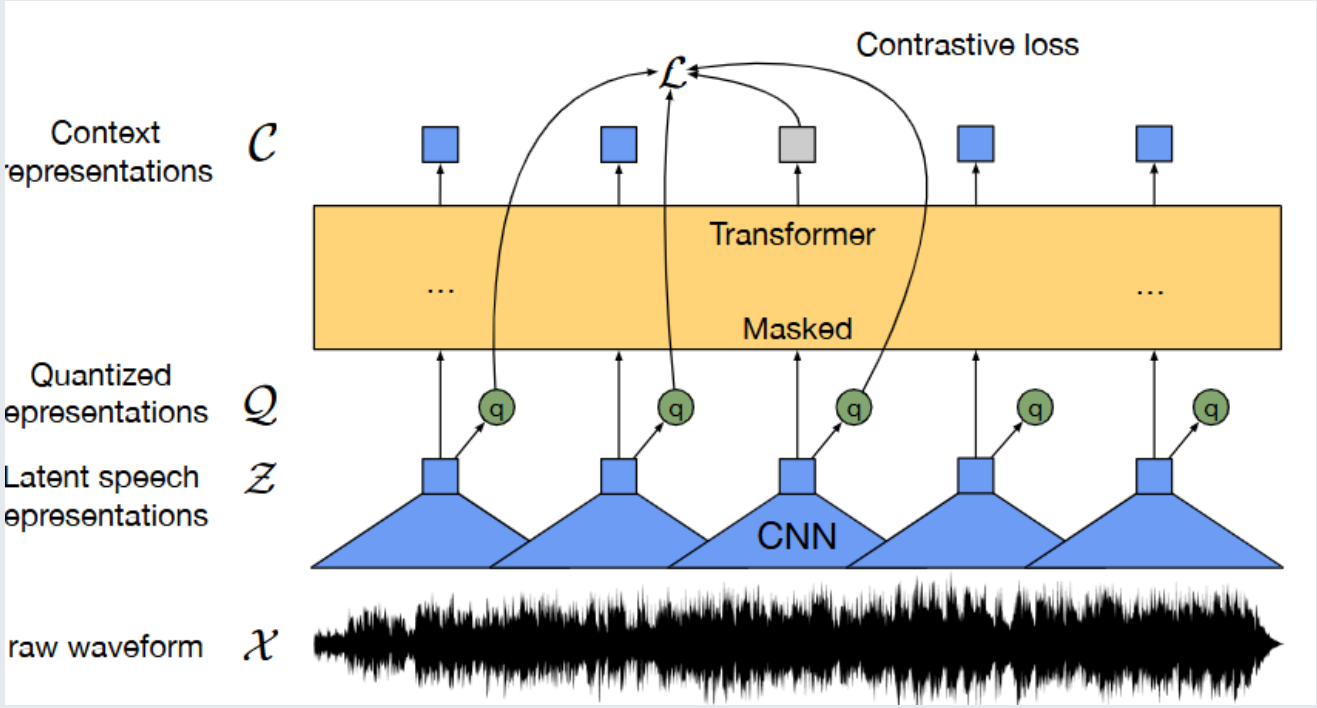
Whisper (French): [‘<|startoftranscript|>', ‘<|fr|>', ‘<|transcribe|>', ‘<|notimestamps|>', ' Comme', ' moi', ',', ' vous', ' avez', ' peut', '-', 'être', ' déjà', ' v', 'éc', 'u', ' cette', ' scène', '.', ' Qu', "'", 'est', '-', 'ce', ' que', ' tu', ' fais', ' cet', ' été', ' toi', ' ?', ' É', 'c', 'oute', ',', ' je', ' ne', ' sais', ' pas', ' encore', ' trop', ……. ‘<|endoftext|>’]]

Whisper (Translation): [‘<|startoftranscript|>', ‘<|fr|>', ‘<|translate|>', ‘<|notimestamps|>', ' Like', ' me', ',', ' you', ' may', ' have', ' already', ' experienced', ' this', ' scene', '.', ' What', ' are', ' you', ' doing', ' this', ' summer', '?', ' I', ' don', "'t", ' know', ' yet', ……. '<|endoftext|>']

Estado del arte: Wav2Vec (Meta)

Modelo que aprende representaciones del habla de forma auto-supervisada, es decir, aprende a predecir unidades enmascaradas del habla, en vez de aprender a partir de datos etiquetados.

Es especialmente útil para la transcripción de lenguajes minoritarios donde hay una cantidad muy limitada de datos etiquetados.



Arquitectura del modelo Wav2vec. Extraído de https://huggingface.co/blog/fine-tune-wav2vec2-english

ASR en la práctica

ASR en la práctica: Herramientas

- **Servicios en la nube:** APIs web donde se envía el audio y se recibe texto transcrito de alta calidad. Implican costo y hay que considerar demoras por latencia (*Google Speech-to-text, Amazon Transcribe, Azure Speech*).
- **Herramientas open-source:** Repositorios de GitHub o frameworks que proveen de los modelos necesarios para realizar la tarea de STT (*DeepSpeech, SpeechBrain, HuggingFace Transformers*)
- **Whisper de OpenAI:** Mención especial a Whisper por su facilidad de uso gracias a que han desarrollado una biblioteca nativa de Python (openai-whisper).

ASR en la práctica: Herramientas

- **Servici** texto transc noras por lat).
- **Herram** que provee (*DeepS*
- **Whisp** e uso gracias enai-whisp

Google Speech-to-Text

amazon TRANSCRIBE

Azure Speech

SpeechBrain

DeepSpeech

Hugging Face

ASR en la práctica: Consideraciones I

- 🔊 **Calidad de audio:** Un audio claro, con buen volumen y poco ruido, dará mucho mejor resultado. Muchos sistemas implementan filtrado de ruido antes de la transcripción.
- 🔷 **Tiempo real vs batch:** Para interfaces interactivas, a veces se necesita transcripción palabra por palabra mientras el usuario habla (p.ej., subtítulos en directo). Técnicas como *endpointing* (detectar pausas para saber cuándo terminar de transcribir) y *streaming* ASR (emitir resultados parciales a medida que llega audio) son importantes.
- 📄 **Puntuación y formato:** La salida de un ASR suele ser texto sin puntuación ni mayúsculas. Algunos avanzados agregan comas y puntos automáticamente usando modelos de puntuación. En otros casos se necesita post-procesar la transcripción (por ejemplo, convertir “siete a m” a “7:00 AM”.

ASR en la práctica: Consideraciones II

- **Métricas:** El estándar es la Word Error Rate (WER), que calcula porcentaje de palabras equivocadas en la transcripción comparado con un "ground truth". Más allá de WER, también es importante la **latencia** (¿tarda medio segundo o 5 segundos en responder el asistente?), y la **robustez** ante diferentes locutores.
- **Entrenamiento personalizado:** Si tu aplicación es muy específica (ej: reconocimiento de comandos en entornos industriales con ruido muy particular), podrías necesitar entrenar o afinar un modelo con datos propios.

Referencias

- [1] Hidden Markov Models (2025). ScienceDirect. https://www.sciencedirect.com/topics/neuroscience/hidden-markov-model
- [2] Mael Fabien (2020, Mayo 26) Introduction to Automatic Speech Recognition (ASR). https://maelfabien.github.io/machinelearning/speech_reco/#
- [3] Jurafsky, D., & Martin, J. H. (2025). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/
- [4] Matt Payne (2021, Sept 29). What is Beam Search? Explaining The Beam Search Algorithm. Width.ai. https://www.width.ai/post/what-is-beam-search
- [5] Andrew Maas (2025). Spoken Language Processing (CS224S). Stanford. https://web.stanford.edu/class/cs224s/semesters/2025-spring/syllabus
- [6] Mamyrbayev Orken et al. (2022). A study of transformer-based end-to-end speech recognition system for Kazakh language. Scientific Reports. https://doi.org/10.1038/s41598-022-12260-y
- [7] Alexei Baevski et al. (2020). Wav2vec 2.0: A framework for Self-Supevised Learning of Speech Representations. https://arxiv.org/pdf/2006.11477
- [8] Loren Lugosch (2020). Sequence-to-sequence learning with Transducers. https://lorenlugosch.github.io/posts/2020/11/transducer/

Referencias

- [9] Abdoli, Sajjad & Cardinal, Patrick & Koerich, Alessandro. (2019). End-to-End Environmental Sound Classification using a 1D Convolutional Neural Network. Expert Systems with Applications. https://www.sciencedirect.com/science/article/pii/S0957417419304403