abril, 2024

Modelos de reconocimiento del habla de código abierto
Tecnologías del habla

Alberto García Martín

VNiVERSiDAD D SALAMANCA

Máster en Sistemas Inteligentes
Universidad de Salamanca

Índice

1. Introducción

Esta memoria tiene como objetivo recoger el trabajo realizado para la asignatura de Tecnologías del habla del Máster en Sistemas Inteligentes de la Universidad de Salamanca. En ella se recoge el desarrollo de una herramienta de reconocimiento del habla de código abierto, así como la revisión del estado del arte en este campo, centrándose en los modelos de reconocimiento del habla de código abierto.

En la última década, el reconocimiento automático del habla ha experimentado un avance significativo gracias al desarrollo de técnicas de aprendizaje profundo y al aumento de la capacidad computacional. Estos avances han permitido la creación de sistemas más precisos y robustos, capaces de transcribir el habla en una amplia variedad de contextos y dominios. Sin embargo, muchos de estos sistemas son propietarios y no están disponibles para el público en general, lo que limita su accesibilidad y su potencial para ser adaptados a nuevas aplicaciones.

En este contexto, el surgimiento de modelos de reconocimiento del habla de código abierto ha supuesto un hito importante en este campo. Estos modelos permiten democratizar el acceso a esta tecnología, facilitando su uso y su desarrollo por parte de la comunidad científica y tecnológica. Cualquier persona con los recursos computacionales necesarios puede ejecutar uno de estos modelos y adaptarlo a sus necesidades específicas, lo que ha dado lugar a una explosión de servicios ofertados con y sin ánimo de lucro.

Sin embargo, es preciso tener en cuenta que no todos los modelos de código abierto son igual de transparentes. Algunos no van más a allá de publicar los parámetros de sus modelos, sin proporcionar información sobre cómo se han entrenado o cómo se han evaluado. Otros, en cambio, ofrecen una descripción detallada de su arquitectura y de su rendimiento, lo que facilita su reproducción y su comparación con otros modelos.

Con el objetivo de demostrar las capacidades y potencial de estos sistemas de reconocimiento del habla, se ha desarrollado una herramienta que permite transcribir archivos de audio en un tiempo inferior al que dura el propio audio. Esta herramienta se basa en el modelo Whisper de OpenAI [4], uno de los primeros modelos de reconocimiento del habla de código abierto que ha demostrado un rendimiento comparable al de los sistemas propietarios.

Esta herramienta no solo se ha desarrollado con el objetivo de proporcionar resultados cualitativamente buenos, sino también de poder ser ejecutada en un ordenador personal sin necesidad de recurrir a recursos computacionales costosos. Además, la latencia de la respuesta ha de ser lo suficientemente baja como para poder ser utilizada en tiempo real, por lo que se tendrá en cuenta el tamaño de los distintos modelos y la velocidad de inferencia de las distintas implementaciones.

2. Revisión del estado del arte

Este apartado se centrará en revisar los modelos de reconocimiento del habla de código abierto que hoy en día obtienen los mejores resultados. Una característica común de estos modelos es que se basan en arquitecturas de aprendizaje profundo, como las redes neuronales recurrentes o convolucionales. Aunque si hay una arquitectura que destaca por encima de las demás, esa es la de los transformers, que han demostrado ser especialmente efectivos en tareas de procesamiento del lenguaje natural.

Para reducir el ámbito de la revisión, se han seleccionado los modelos que han sido publicados en los últimos dos años y que han obtenido los mejores resultados en los benchmarks de referencia. Además, se han tenido en cuenta aquellos modelos que han sido liberados bajo una licencia de código abierto, que permita su uso y su modificación por parte de terceros.

Estos modelos son Whisper de OpenAI [4], SeamlessM4T de Meta [2], y Canary de NVIDIA [3]. No es de extrañar que estos modelos hayan sido desarrollados por grandes empresas tecnológicas, ya que entrenan sus modelos con una gran cantidad de datos y recursos computacionales, lo que les permite obtener resultados de alta calidad. Hay que puntualizar que de estos modelos, solo Whisper se consideraría un modelo de código abierto bajo la definición de la Open Source Initiative, ya que los otros dos modelos tienen restricciones en su uso con fines comerciales y técnicamente serían considerados software de código "disponible".

Todos ellos tienen bastantes características en común en cuanto a su arquitectura, además del hecho de que han sido entrenados con grandes cantidades de datos y que emplean bloques de transformers. Los tres toman como entrada un espectrograma con las frecuencias ajustadas a la escala de Mel, esta escala es una transformación no lineal de la frecuencia que se ajusta mejor a la percepción humana del sonido. En esta escala cada tono se percibe como equidistante de los demás, lo que facilita la extracción de características relevantes para el reconocimiento del habla.

Además, todos estos modelos son modelos de reconocimiento del habla de extremo a extremo, distinguiéndose de los modelos tradicionales que contenían múltiples componentes, como un modelo acústico, un modelo de lenguaje, etc. En estos modelos, la entrada es el espectrograma y la salida es el texto transcribiendo el habla. Esto permite que aprendan directamente la correspondencia entre la señal de audio y la transcripción textual, simplificando el proceso de entrenamiento y facilitando su adaptación a nuevos lenguajes.

2.1. Whisper

Whisper es un modelo publicado originalmente en 2022, su arquitectura se puede observar en la figura 1. El espectrograma que se introduce como entrada (de 30 segundos como máximo) se pasa por un red neuronal convolucional para extraer características relevantes, posteriormente se tiene en cuenta el posicionamiento

sinusoide para añadir información sobre la posición de las características extraídas. A continuación, se pasa por un codificador y un decodificador de transformers, que se encargan de aprender y predecir el siguiente token en la secuencia de texto.

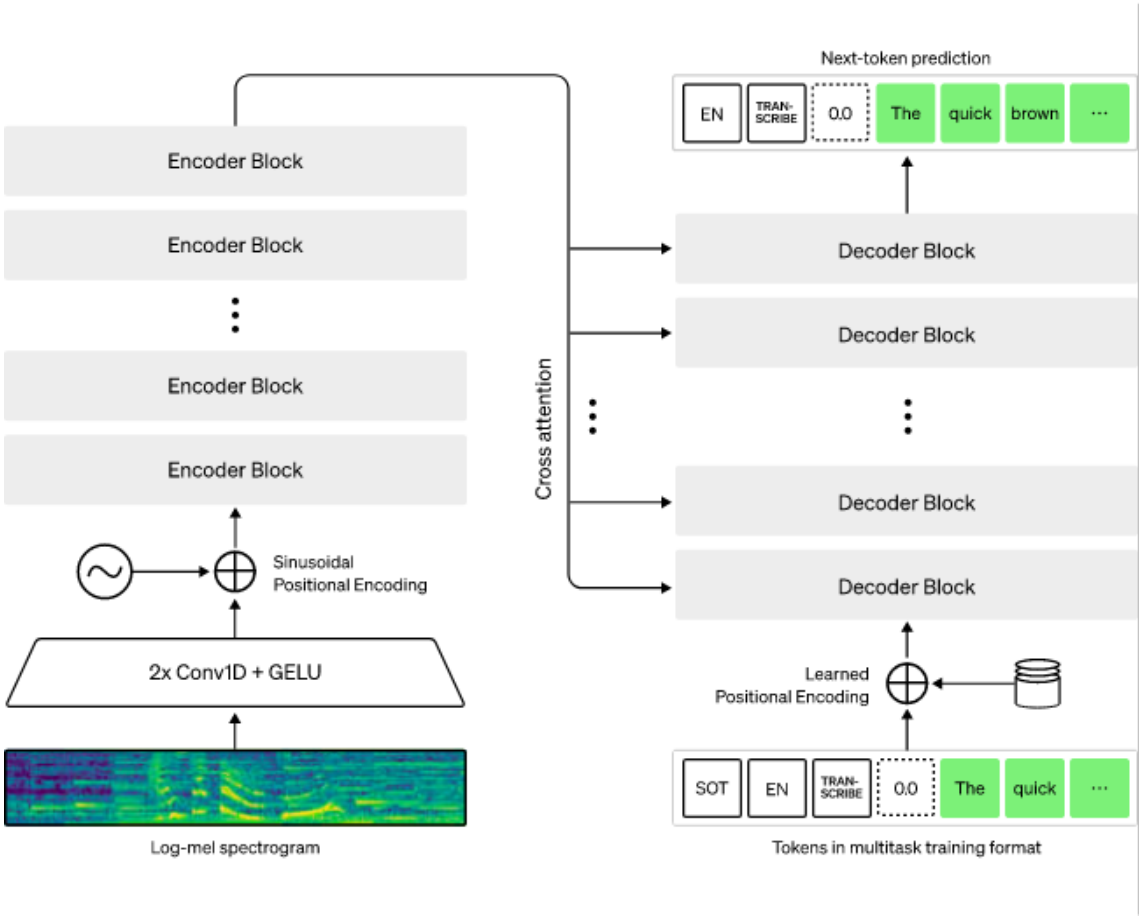

Figura 1: Arquitectura de Whisper

La mayor fortaleza de Whisper cuando se publicó fue su capacidad para transcribir diálogos a la primera sin necesidad de adaptación, a diferencia de otros modelos que requerían de un ajuste para obtener buenos resultados. Además, fue entrenado con un conjunto de datos multilingüe, por lo que, aunque principalmente se entrenó con datos en inglés, también es capaz de transcribir otros idiomas con un rendimiento aceptable.

OpenAI no solo libero un único modelo preentrenado, sino que publicó una familia de modelos con distinta cantidad de parámetros, desde el modelo más pequeño con 39 millones de parámetros hasta el modelo más grande con 1550 millones de parámetros. Esto permite a los usuarios del modelo elegir aquel que mejor se ajuste a sus necesidades, ya sea por la velocidad de inferencia, la precisión o la cantidad de recursos requeridos. En la tabla 1 se pueden ver las características de los distintos modelos de la familia Whisper.

También se han publicado versiones de Whisper posteriores a la original, que han mejorado su rendimiento en distintos aspectos. Whisper 2 fue entrenado durante 2.5

épocas más que Whisper. Whisper 3, por otro lado, fue entrenado con un conjunto de datos más grande, que incluía datos etiquetados por Whisper 2, esto permitió mejorar su rendimiento entre un 10 % y un 20 % en distintas métricas.

Modelo	Capas	Dimensión	Cabezas	Parametros
Tiny	4	384	6	39M
Base	6	512	8	74M
Small	12	768	12	244M
Medium	24	1024	16	769M
Large	32	1280	20	1550M

Tabla 1: Modelos de la familia Whisper

2.2. SeamlessM4T

Este modelo fue publicado por Meta en 2023, a diferencia de Whisper, que se centraba principalmente en la tarea de reconocimiento automático del habla, SeamlessM4T es un modelo multitarea que ha sido entrenado para realizar tareas de reconocimiento del habla, síntesis de voz, traducción automática de voz y texto. También está más centrado en su capacidad multilingüe, como representa el diagrama de la figura 2, es capaz de detectar segmentos de distintos idiomas en un mismo audio y transcribirlos en el idioma correspondiente.

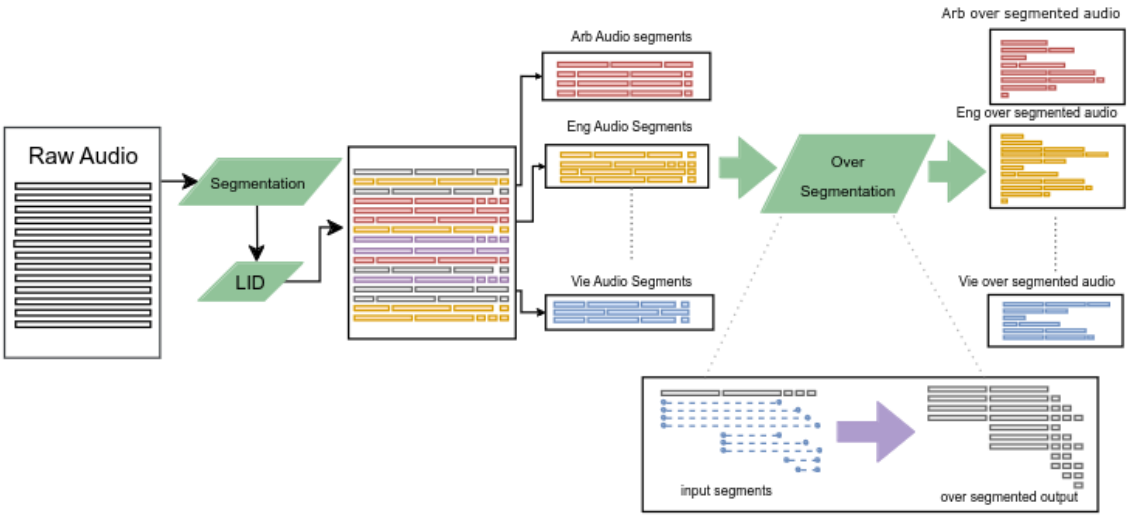

Figura 2: Procesamiento del habla en SeamlessM4T

Unos meses más tarde de la publicación de SeamlessM4T, Meta publicó SeamlessM4T v2, una versión mejorada del modelo original con cambios en su arquitectura y en su entrenamiento, mejorando principalmente su rendimiento en tareas multilenguaje.

Debido a su enfoque multitarea, SeamlessM4T es un modelo con una arquitectura más compleja que la de Whisper. Como se puede ver en la figura 3, está formado por cuatro modulos principales, divididos por colores en la figura, que se encargan de realizar las distintas tareas. Cada uno de estos módulos tiene su propia arquitectura y sus propios parámetros, lo que permite que el modelo pueda ser entrenado de forma independiente en cada tarea. El bloque azul es un transformer que se encarga de la codificación y decodificación del texto, el naranja es un conformer que codifica el habla que pasa al transformer, el morado se encarga de generar unidades fonéticas a partir del texto y el verde es una red GAN que se encarga de la síntesis final de voz.

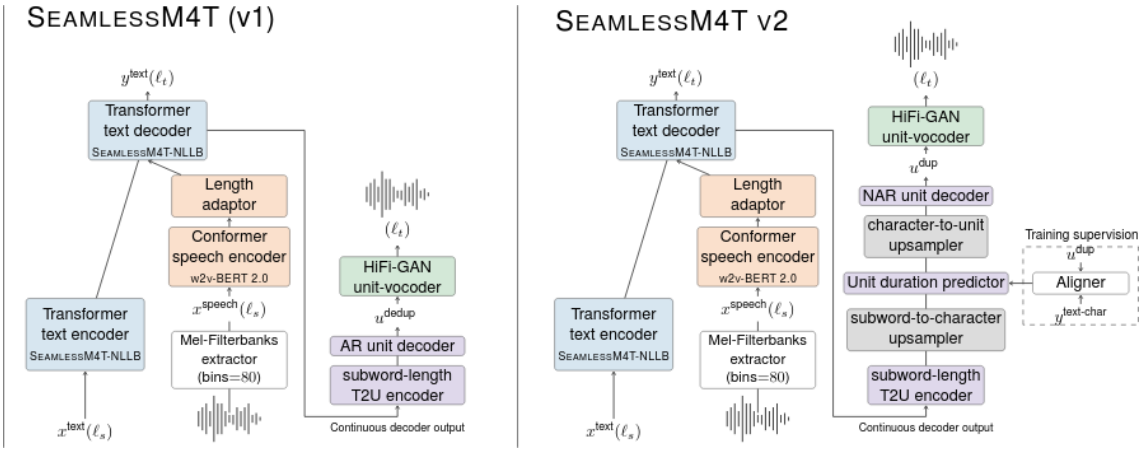

Figura 3: Arquitectura de SeamlessM4T v1 y v2

El tamaño de SeamlessM4T también es considerablemente mayor que el de Whisper, con 2300M de parámetros en la versión "large" y 1200M en la versión "medium". Esto se debe a que el modelo tiene que aprender a realizar tareas adicionales, como la traducción automática de voz y texto, que requieren de una mayor capacidad de representación.

2.3. Canary

Canary es un modelo publicado por NVIDIA a inicios de 2024, a diferencia de los otros dos anteriores, este no fue acompañado por un artículo científico que describiera su arquitectura y su entrenamiento, tan solo se publicó un comunicado de prensa. Este modelo no se centra tanto en la capacidad multilingüe, sino que reduce su ámbito a la transcripción de audio de los principales idiomas occidentales, el inglés, el español, el alemán y el francés.

Este modelo ha superado a Whisper en distintos benchmarks de referencia en cuanto a reconocimiento del habla en inglés y en los otros idiomas mencionados. Por ejemplo, en el conjunto de test del Mozilla Common Voice [1] para los idiomas indicados, Canary ha obtenido una tasa de error de palabra (WER) inferior a los dos modelos descritos anteriormente, los resultados se pueden ver en la figura 4.

Los escuetos detalles sobre la arquitectura del modelo apuntan a una arquitectura muy similar a la de Whisper, con un codificador y un decodificador de transformers,

aunque no se ha especificado el número de capas, cabezas o dimensiones. Sí que se ha especificado que el modelo tiene 1000M de parámetros, lo que lo sitúa por debajo de SeamlessM4T y por debajo de Whisper en su versión más grande, aunque Canary no ofrece versiones más pequeñas, a diferencia de Whisper.

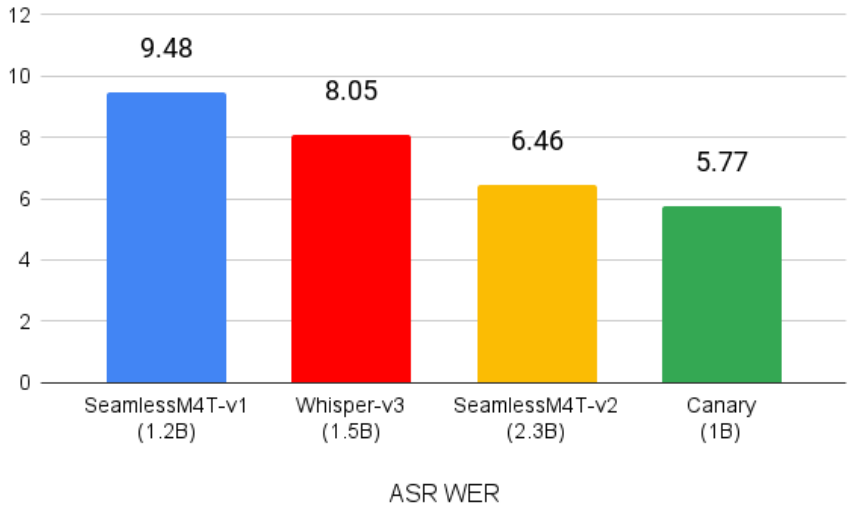

Figura 4: WER medio en el Mozilla Common Voice

Canary también ha conseguido posicionarse en primer lugar en el Open ASR Leaderboard [5] llevado acabo por Hugging Face. Esta clasificación es una agregación de los resultados obtenidos por distintos modelos abiertos, en distintos benchmarks de reconocimiento del habla, como LibriSpeech, Common Voice 9, GigaSpeech, etc. Aunque solo se centra en el inglés. En la figura **??** se puede ver la posición de Canary en este ranking.

Leaderboard | Metrics | Request a model here!

model	Average WER	RTF (1e-3)	AMI	Earnings22	Gigaspeech	LS Clean
nvidia/canary-1b	6.67	9.8	14	12.25	10.19	1.49
nvidia/parakeet-tdt-1.1b	6.95	1.7	15.9	14.65	9.55	1.39
nvidia/parakeet-rnnt-1.1b	7.04	2.4	17.1	14.15	9.96	1.46
nvidia/parakeet-ctc-1.1b	7.58	0.6	15.6	13.69	10.27	1.83
nvidia/parakeet-rnnt-0.6b	7.63	2	17.56	14.9	10.07	1.62
openai/whisper-large-v3	7.7	7.45	16.01	11.3	10.02	2.03

Figura 5: Open ASR Leaderboard

Mide la precisión de los modelos mediante la métrica de WER, que mide el porcentaje de palabras que se han transcritos incorrectamente a partir de la distancia de Levenshtein entre la transcripción del modelo y la transcripción de referencia. Además también indica el factor de tiempo real (RTF) que mide la velocidad de inferencia del modelo, un RTF de 1 indica que el modelo tarda el mismo tiempo en transcribir un audio que dura el audio, un RTF de 0.5 indica que el modelo tarda la mitad de tiempo, etc.

3. Desarrollo de la aplicación

Para demostrar las capacidades de los modelos de reconocimiento del habla de código abierto, se ha desarrollado una aplicación que permite transcribir archivos de audio en tiempo real. Esta aplicación se basa en el modelo Whisper de OpenAI.

El criterio tomado para seleccionar Whisper como modelo base de la aplicación ha sido su rendimiento en distintos benchmarks de referencia, y su oferta de modelos preentrenados con distintos tamaños y capacidades, lo que permite adaptar el modelo a las necesidades específicas de la aplicación. A pesar de que Canary y SeamlessM4T han obtenido resultados superiores en algunos benchmarks, solo ofrecen versiones de un tamaño igual o superior a 1000M de parámetros, lo que requiere de una capacidad de memoria de entre 2 y 4 GB para poder ejecutar el modelo, según la precisión de coma flotante que se utilice. Estos requisitos excluirían a muchos usuarios de poder ejecutar la aplicación en sus ordenadores personales.

Mientras tanto, el modelo más pequeño de Whisper solo tiene 39M de parámetros, requiriendo tan solo de 30 a 70 MB de memoria para su ejecución. Además, existen múltiples implementaciones de Whisper que optimizan el código original para mejorar la velocidad de inferencia, sin perder precisión en la transcripción. Por lo tanto, actualmente Whisper es el modelo más adecuado para ser utilizado en una aplicación de reconocimiento del habla en tiempo real.

En esta aplicación se ha utilizado whisper.cpp [1] que es una implementación de Whisper en C/C++ optimizada para la velocidad de inferencia. Además también permite hacer uso de la GPU para acelerar la inferencia en sistemas con el hardware adecuado. Whisper.cpp proporciona bindings para Go, por lo que se ha desarrollado una aplicación con interfaz gráfica empleando el framework Wails [2], que permite desarrollar aplicaciones de escritorio multiplataforma con Go y tecnologías web, facilitando el desarrollo comparado con alternativas basadas en C/C++ sin perder rendimiento.

El *frontend* de la aplicación se ha desarrollado con Svelte y Tailwind CSS, que permiten desarrollar interfaces de usuario interactivas y atractivas de forma sencilla. La aplicación permite cargar un archivo de audio en formato WAV y con una frecuencia de muestreo de 16 kHz. Una vez cargado el archivo, se puede reproducir el audio y transcribirlo. En la figura 6 se puede ver una captura de pantalla de la aplicación.

La aplicación viene por defecto empaquetada con el modelo Whisper Tiny cuantizado a 5 bits, siendo el modelo más pequeño y rápido de la familia Whisper, pesando tan solo 30.7 MB. Aunque también se pueden descargar el resto de modelos de la familia Whisper desde la aplicación, para poder utilizarlos en la transcripción, la lista de modelos disponibles se puede ver en la figura 7. En caso de tener una GPU de NVIDIA, se puede hacer uso de ella recompilando la aplicación con la variable de entorno `WHISPER_CUDA=1`.

[1] `https://github.com/ggerganov/whisper.cpp`
[2] `https://wails.io/`

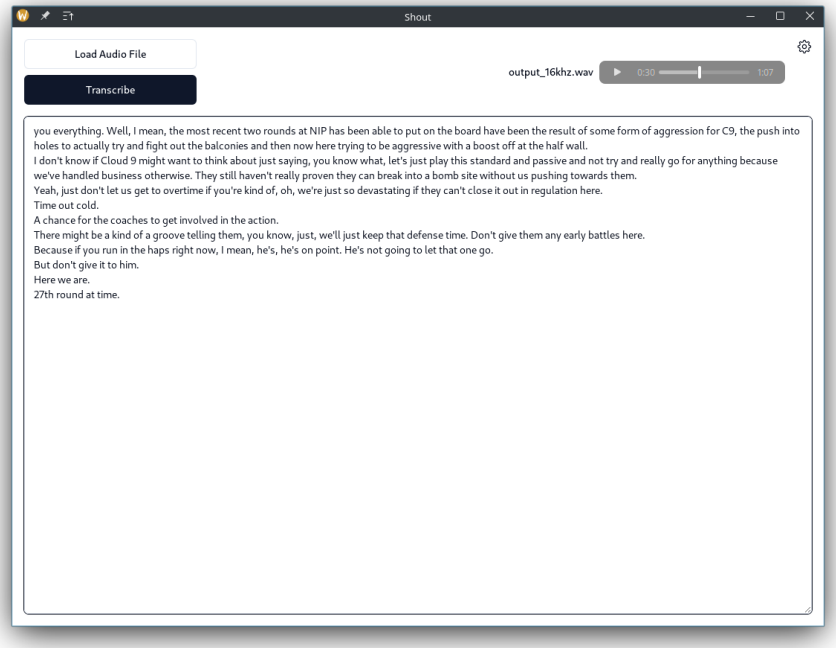

Figura 6: Captura de pantalla de la aplicación

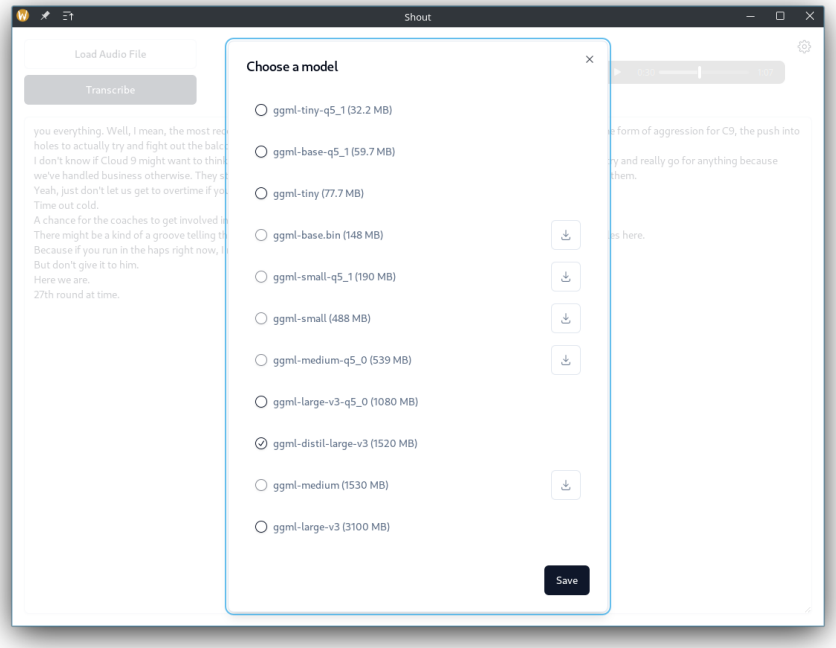

Figura 7: Configuración de la aplicación

El *backend* de la aplicación está dividido en cuatro módulos principales, *main.go* que se encarga de la inicialización de la aplicación y del exponer los archivos locales necesarios para la interfaz web a través de un servidor HTTP. *app.go* se encarga de

cargar y guardar los ajustes de la aplicación, de cargar el modelo seleccionado en memoria y de exponer las funciones que se pueden llamar desde el *frontend*. *download.go* se encarga de descargar los modelos de la familia Whisper desde los servidores de Hugging Face, y por último *process.go* se encarga de la propia transcripción del audio, decodificando el audio en formato WAV y enviándolo a Whisper para obtener la transcripción.

Todas estas operaciones se realizan de forma asíncrona y en hilos separados, para no bloquear la interfaz de usuario y para poder transcribir el audio sin interrupciones. De esta manera se consigue una velocidad de transcripción muy superior a la duración del audio, permitiendo transcribir un audio de más de un minuto en menos de 10 segundos.

4. Conclusiones

En este trabajo se ha revisado el estado del arte en modelos de reconocimiento del habla de código abierto, estudiando detalles sobre su arquitectura y su rendimiento. Descubriendo que en muchos casos, estos modelos superan a los modelos propietarios en distintos benchmarks de referencia.

Además, se ha desarrollado una aplicación de reconocimiento del habla que se ejecuta de forma local y que permite transcribir archivos de audio sin necesidad de conexión a internet o dependencia de servicios de terceros. Ha sido un trabajo muy interesante que ha permitido aprender como funcionan los modelos punteros en este campo y también como usarlos en una aplicación real.

Aunque también hay que asumir que el desarrollo de la aplicación ha tenido sus dificultades, sobre todo con la integración de whisper.cpp en la aplicación y con la comunicación entre las distintas partes de la aplicación. A pesar de ello, se ha logrado desarrollar una aplicación funcional que cumple el objetivo de transcribir archivos de audio en tiempo real. Aún quedan aspectos por mejorar, como el soporte para más formatos de audio, que me gustaría afrontar en el futuro a nivel personal.

Junto con esta memoria se ha entregado el código fuente de la aplicación, que puede ser compilado en cualquier sistema operativo que tenga instalado las dependencias de Wails ejecutando el comando `make` en la carpeta del proyecto. Se ha comprobado el correcto funcionamiento de la aplicación en un sistema con Fedora 39 y en otro con Ubuntu 22.04.

Referencias

[1] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers y Gregor Weber. Common Voice: A Massively-Multilingual Speech Corpus. 5 de marzo de

2020. DOI: 10.48550/arXiv.1912.06670. URL: http://arxiv.org/abs/1912.06670.

[2] Seamless Communication, Loic Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan y John Hoffman. SeamlessM4T: Massively Multilingual & Multimodal Machine Translation. 24 de octubre de 2023. DOI: 10.48550/arXiv.2308.11596. URL: http://arxiv.org/abs/2308.11596.

[3] Krishna C. Puvvada, Piotr Zelasko, He Huang, Oleksii Hrinchuk, Nithin Rao Koluguri y Somshubra Majumdar. NVIDIA NeMo Canary. URL: https://nvidia.github.io/NeMo/blogs/2024/2024-02-canary/.

[4] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey e Ilya Sutskever. Robust Speech Recognition via Large-Scale Weak Supervision. Versión 1, 2022. DOI: 10.48550/ARXIV.2212.04356. URL: https://arxiv.org/abs/2212.04356.

[5] Vaibhav Srivastav, Somshubra Majumdar, Nithin Koluguri, Adel Moumen, Sanchit Gandhi, Hugging Face Team, Nvidia NeMo Team y SpeechBrain Team. Open Automatic Speech Recognition Leaderboard. https://huggingface.co/spaces/hf-audio/open_asr_leaderboard, 2023.