

Separación de fuentes auditivas para pedagogía musical

Sound source separation for musical pedagogy

Randy Darrell Lancheros-Molano¹ , Juan Sebastián Triana-Perez¹ , Juan Felipe Castañeda-Chaparro¹ , Felipe Andrés Gutiérrez-Naranjo¹ , Andrea del Pilar Rueda-Olarte¹ 

¹Pontificia Universidad Javeriana, Bogotá D.C., Colombia
randy.lancheros@javeriana.edu.co, juan triana@javeriana.edu.co, castaneda_j@javeriana.edu.co,
felipe_gutierrez@javeriana.edu.co, andrea@javeriana.edu.co

(Recibido: 20 octubre 2020; aceptado: 15 abril 2021; Publicado en Internet: 1 junio 2021)

Resumen. *Harmonics* espera apoyar a la pedagogía musical, ofreciendo un producto concreto con el cual los interesados en aprender a tocar un instrumento puedan practicar. Se entrenó un modelo para identificar y aislar las pistas singulares de una canción, por medio de *TensorFlow* y herramientas para realizar la separación de fuentes auditivas y producir partituras genuinas, basadas en un algoritmo de transcripción musical (para pianos, bajos, batería y voz, específicamente), que los principiantes puedan visualizar, editar y descargar (en formatos .PDF y .MIDI), ajustándose a su ritmo de práctica. Se consideraron tres métodos de separación de fuentes, bajo las siguientes restricciones: emplear una única canción como archivo de entrada, que ésta fuera moderadamente compleja (compuesta por un conjunto de entre tres y seis instrumentos) y que la cantidad de muestras –canciones compuestas por instrumentos relevantes y pistas de cada instrumento por separado– aptas para el entrenamiento del modelo, sean sumamente escasas.

Palabras clave: Aprendizaje de máquina, Separación de fuentes auditivas, Generación de partituras, Aplicación web.

Abstract. *Harmonics* hopes to support musical pedagogy, offering a concrete product with which those interested in learning to play an instrument can practice. We trained a model to identify and isolate the singular tracks of a song through TensorFlow and tools to make the separation of auditory sources and produce genuine sheet music, based on a musical transcription algorithm (specifically for pianos, basses, drums, and voice) that beginners can visualize, edit, and download (in .PDF and .MIDI formats), adjusting at their own pace. Three methods of source separation were considered, under the following restrictions: Use a single song as an input file, which it was moderately complex (composed of a set of between three and six instruments), and that the number of samples –songs composed by relevant instruments and tracks of each standalone instrument – suitable for model training, would be extremely scarce.

Keywords: Machine learning, Sound source separation, Sheet music generation, Web application.

Tipo de artículo: Artículo de investigación.

1 Introducción

La representación digital de una canción es considerablemente compleja: varios instrumentos se ven involucrados, junto a filtros y reverberaciones, donde los mismos parámetros de producción cambian continuamente (Cano et al., 2019). Como se requiere de un elemento concreto de apoyo para el estudio, se necesita una herramienta capaz de lidiar con esta complejidad, al centrarse en un instrumento específico que se desee aprender, con el fin de aislar tal pista rápida y cómodamente para el usuario. De este modo, se podría trabajar con cada instrumento por separado y enfocarse en generar un producto práctico a partir de cada uno, en pro de la pedagogía.

La investigación en el campo de la neurobiología que designa al ser humano como un ser naturalmente musical ya se conoce (Peretz, 2006). Asimismo, se argumenta que la imitación es un medio para aprender a un ritmo más acelerado, debido a la practicidad del ser humano (Byrne & Russon, 1998). En otro orden de ideas, se ven tecnologías novedosas como el aprendizaje de máquina: un esfuerzo dentro del campo de la Inteligencia Artificial, con el cual dotar a las máquinas de la capacidad de aprender de su entorno (Shinde & Shah, 2018). Al considerar todos estos conceptos a la vez, se vislumbra la oportunidad: valerse de

técnicas de aprendizaje de máquina sobre temas musicales, que le resulten interesantes a un gran grupo de personas, que puedan integrarse con la separación de fuentes, en pro del aprendizaje y la enseñanza de estos temas.

De entre todas los conceptos, técnicas y herramientas dentro del campo de aprendizaje de máquina, el más apropiado para el proyecto fue *Spleeter*. Ésta es una herramienta para la rápida separación de fuentes musicales por medio de modelos pre-entrenados, a base de *U-Net* (arquitectura de codificación y decodificación sobre redes neuronales convolucionales) de forma tal que la separación sobre los espectrogramas estimados resulte efectiva (Hennequin et al., 2020). Esta herramienta fue la inspiración de la base del código de *Harmonics*, la cual permitió el desarrollo hasta la producción de partituras. Las mayores diferencias entre dicha base y *Spleeter* se ven en el uso de modelos propios y en la forma en que se realizó *data augmentation*, al aplicar transformaciones temporales, de afinación, y de efecto de sonidos.

Además de trabajos con productos en mente, también se encuentran aquellos de carácter investigativo. Un ejemplo de estos es la comparación de múltiples decodificadores sobre la separación de fuentes mediante redes neuronales convolucionales, cuyo objetivo radicó en validar el modelo de trabajo bajo los índices de relación: señal a distorsión, señal a artefactos, señal a interferencias, e imagen a distorsión espacial (Pepino & Bender, 2018). Este trabajo es similar a las raíces de *Harmonics* como un proyecto de análisis comparativo de algoritmos de separación de fuentes de audio, antes de que el enfoque se centrara en la creación de un producto concreto para asistir en la pedagogía musical.

Apoyos similares a la pedagogía pueden verse en la herramienta PHENICX, o *Performances as Highly Enriched aNd Interactive Concert eXperiences*, la cual es una herramienta para el análisis de conciertos en vivo, con funcionalidades como la muestra de qué instrumentos están tocando en algún momento dado, o la representación de las tramas aisladas por medio de espectrogramas (Gómez et al., 2013). Ésta es ejemplar de un proyecto al que *Harmonics* espera emular, pero dentro de un alcance mucho más restringido, que se enfoque en personas singulares. Con esto en mente, otras opciones que ayuden a las personas son la pedagogía de música popular, la cual se basa en la improvisación grupal y la práctica autogestionada informal (Lebler, 2008), considerada insuficiente por sí sola, o la creación de una plataforma para dictar clases virtuales, la cual no presenta un desarrollo tecnológico en sí.

Proporcionar tal plataforma, especialmente en el contexto de la pandemia del 2020 e inicios del 2021, presentaría una solución viable y familiar para las personas que deseen practicar música. Sin embargo, la coordinación necesaria con profesionales y la comunidad de aprendices requeriría de un emprendimiento en sí, fuera del alcance del proyecto. Este mismo problema limita la implementación de una clase bajo la pedagogía de música popular. A pesar de esto, el espíritu de práctica y repetición indiscriminada, al gusto del individuo, se rescata al proporcionar las partituras que el usuario desee, para que practique la edición de las mismas y/o tocar el instrumento de su preferencia, a su discreción, sin que tenga que molestarse con horarios, clases o personas a las que dedicarles tiempo. Estas características hacen de *Harmonics* una herramienta apropiada para la pedagogía musical, bajo un nicho de aprendizaje individual que tiene precedente.

Los posibles campos de aplicación de la separación de fuentes son diversos. Ésta puede aplicarse en campos como la música, mejorando las operaciones a realizar sobre la representación digital de una canción (Jouny, 2007). Desde otra perspectiva, se ve cómo aprender a tocar un instrumento, por sí solo, ya trae varios beneficios como el estímulo de la memoria, la paciencia, y la creatividad (Roulston et al., 2015). Visto así, el valor de trabajar con el procesamiento musical es tanto relevante como integral: éste ofrece un tema rico para el desarrollo de un proyecto ingenieril, empleado en contextos que relacionen a la formación profesional con la vida personal.

Al ver la complejidad de la música digital y la falta de un producto práctico y sencillo en las alternativas mencionadas con anterioridad, se propone realizar un algoritmo que tome una canción por entrada y realice un proceso de transcripción musical, para generar una partitura genuina con la que el usuario pueda practicar. Como tal idea, en teoría, podría aplicarse sobre cualquier canción. Se acota el alcance de la propuesta a trabajar con los siguientes instrumentos: piano, bajo, voz y batería.

Este artículo está compuesto por secciones, empezando con la metodología planeada, dividida en cuatro fases: investigación, adaptación, implementación, y pruebas. Luego, se presentan los resultados de las diferentes pruebas que se aplicaron, buscando validar la aceptación de los usuarios y la viabilidad de la aplicación en un entorno de la vida real. Por último, se presentan las conclusiones del proyecto y las oportunidades de trabajo futuro.

2 Metodología

La metodología del proyecto está dividida en cuatro fases: la fase de investigación, donde se exploraron los métodos de separación de fuentes auditivas con los que se podría trabajar; la fase de adaptación, donde se plantea el diseño de *Harmonics* con base en lo hallado en la primera fase; la fase de implementación, donde se describe el proceso de desarrollo de *Harmonics*, y la fase de pruebas, donde se describe cómo se plantearon las pruebas para verificar la aceptabilidad de las partituras.

2.1 Fase de investigación

Esta fase se enfocó en la investigación de los posibles métodos de separación de fuentes que pudieran aplicarse y en las herramientas a utilizar en el desarrollo de la aplicación *Harmonics*. El desarrollo de esta fase se basa en el uso de la metodología Kanban (Agile Alliance, 2021), principalmente por su *Backlog* de tareas, con el cual se asignan las tareas correspondientes a cada integrante del grupo, y la capacidad de paralelizar las fases posteriores dentro de un marco de trabajo ágil.

Los tres métodos de separación que se investigaron fueron el BSS (*Blind Source Separation*), el cual ignora los parámetros del conjunto de entrada y el canal de transmisión por completo (Yu et al., 2013); el SSS (*Supervised Source Separation*), basado en la implementación de “diccionarios” para reconstruir cada pista separada (Duan et al., 2012), y el SCBSS (*Single Channel Blind Source Separation*), basado en los mismos principios que el BSS, pero toma un único archivo de entrada y la idea principal detrás de él radica en explotar la estructura de tiempo inherente a las fuentes, para generar funciones base con las cuales reconstruir las señales originales (He et al., 2018).

Al investigar, se identificó que ninguno de estos métodos era apto para *Harmonics*. El BSS requiere de múltiples versiones del mismo archivo de audio a separar, grabados desde múltiples micrófonos, como entrada para poder aplicarse (Naik & Wang, 2014), el SSS requiere información de entrenamiento de las ondas de audio que se espera separar, incluyendo su comportamiento esperado (Wytock & Kolter, 2014), y muchos de los algoritmos de SCBSS se basan en señales mezcladas artificialmente, donde se conoce la cantidad de señales originales, valor que normalmente se limita a tan solo dos (Gao et al., 2008). Así, se concluyó que el uso de aprendizaje de máquina para preparar diccionarios, entrenados en el contexto en que se desempeñan los instrumentos esperados, pero ignorantes ante género o acompañamientos, sería la aproximación apropiada para realizar la separación.

2.2 Fase de adaptación

Con el enfoque en aprendizaje de máquina en mente, la siguiente necesidad era la del enfoque de aplicación. Al esperar brindarle una alta disponibilidad al usuario, de forma que acceda a la funcionalidad de la aplicación rápido y sin mayor dificultad, se optó por un desarrollo web. Así, se planteó una división estándar entre *back end* y *front end*, con los siguientes criterios: baja curva de aprendizaje, experiencia previa con las herramientas, alta integrabilidad, una estructura en componentes, y alto rendimiento.

Diseño del sistema. A continuación, se presenta el diagrama de componentes de la aplicación *Harmonics*, ilustrando todos los elementos necesarios para implementar la lógica de negocio en el *back end*, en los módulos de proyectos, usuarios, separación y transcripción, junto a los integradores y aplicaciones de terceros necesarios para la generación de contenido multimedia. De igual manera, se observa la organización de los componentes realizados en el *front end*, tras agrupar el gran conjunto de tecnologías distintas que fueron necesarias para desarrollarlo (ver [Figura 1](#)).

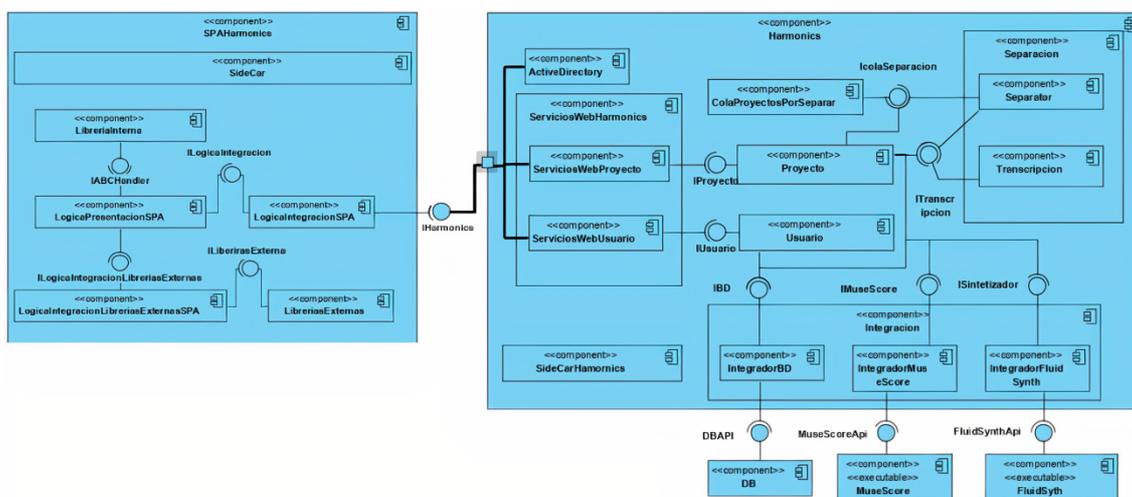


Figura 1. Diagrama de componentes del sistema *Harmonics*

Estos componentes se definen a continuación:

- **ActiveDirectory:** Maneja las sesiones iniciadas por los usuarios.
- **ServiciosWebHarmonics:** Publica las funcionalidades del servidor, por medio de direcciones URL, y recibe las diferentes peticiones de los usuarios.
- **ServiciosWebProyecto y ServiciosWebUsuario:** Exponen todas las funcionalidades que ofrece el componente de proyecto y el de usuario, respectivamente, por medio de servicios REST/HTTP.
- **Proyecto y Usuario:** Implementan la lógica de negocio asociada a la gestión de proyectos y usuarios, respectivamente.
- **ColaProyectosPorSeparar:** Agrega y extrae las tareas (proyectos) en la cola de mensajes.
- **Separación:** Contiene la lógica para la separación de voces y la transcripción sobre las mismas.
- **Separator:** Implementa la funcionalidad de extraer las voces de una canción.
- **Transcripción:** Implementa la funcionalidad de transcribir archivos MIDI a notación musical en PDF, MP3, y notación ABC.
- **IntegradorDB:** Se encarga de la conexión e intercambio de datos con la base de datos.
- **IntegradorMuseScore e IntegradorFluidSynth:** Se encargan del envío e intercambio de datos con la aplicación *MuseScore* y *FluidSynth*, respectivamente.
- **LibreriaInterna:** Implementa la lógica relacionada a la adición de la partitura en la notación ABC, además de la validación de reglas musicales.
- **LogicaPresentacionSpa:** Implementa las distintas interfaces gráficas con las que el usuario interactúa, además de la lógica relacionada a los eventos presentes en ellas.
- **LogicaIntegracionSPA:** Consume los servicios expuestos en el servidor.
- **LogicaInteraccionLibreriasExternasSPA:** Integra las librerías externas utilizadas en el desarrollo de la aplicación web.
- **LibreriasExternas:** Almacena la lógica relacionada a las distintas librerías externas utilizadas en el desarrollo de la aplicación web.
- **SideCar:** Unidad de despliegue interna en cada componente, utilizada para el intercambio de datos.

2.3 Fase de implementación

Esta fase contiene el proceso de desarrollo de *Harmonics*.

Desarrollo del *front end*. El repositorio fue lo primero creado, el cual permitiría el versionamiento del código desarrollado. Al no saber mucho sobre las herramientas con las cuales se desarrollaría la aplicación web, se comenzó investigando cada opción prometedora, con el fin de adquirir el suficiente conocimiento para el desarrollo del proyecto. Tras centrarse en tres posibles candidatos, se realizó un diseño que cumpliera con los requisitos funcionales y estéticos solicitados.

Al adquirir suficiente conocimiento en *React*, se empezaron las interfaces de las pantallas propuestas en el diagrama de componentes. Una vez se concluyó con el diseño GUI en sí, se desarrollaron los controladores de eventos de cada una de estas pantallas, con el fin de dinamizar y renderizar las páginas. Las primeras pruebas realizadas fueron locales, sin integración con el *back end*, validando que se cumpliera con la funcionalidad y estructura esperada. Después, se realizaron las pantallas más sencillas, como *home* y nosotros, junto a la lógica de navegación.

Al finalizar, se le asignaron direcciones URL específicas a cada una de ellas y se mapeó la navegación mediante botones. Cabe resaltar que, en este proceso de comunicación, se verificó que cada pantalla recibiera los datos correctos para inicializarse. El paso siguiente fue realizar los distintos *proxies* que se encargarían de consumir los servicios expuestos por el *back end*. Una vez completados, se realizaron pruebas que permitieran validar el desarrollo hecho y la información recibida por parte del *back end*. Posteriormente, el equipo se dividió en dos, donde una parte se enfocó en desarrollar el editor de partituras exclusivamente, mientras que los demás se enfocaron en crear las distintas imágenes, textos, y ayudas que dotarían a la aplicación de mayor usabilidad.

Hay tres pantallas clave, de mayor complejidad lógica e interacción con el *back end*: “Proyectos” permite la búsqueda filtrada de proyectos, paginando la lista de resultados; “Ítem proyecto” despliega el resultado detallado de una separación dada, permitiendo la interacción con cada archivo multimedia –mediante herramientas embebidas como reproductores para los .MP3 y MIDI, junto a un visualizador .PDF para las partituras– por separado, por medio de una pestaña tabulada, que ofrece la posibilidad de descarga para el proyecto completo, o para una voz específica; “Editor de partituras” ofrece varias opciones de edición gracias a *ABCHandler*, que por cuestiones de tiempo, alcance y diseño, se limitaron a las principales –cambios de clave, compás, tonalidad, junto a adición y borrado de notas y silencios– además de un reproductor MIDI para la partitura, utilizable tras cada cambio realizado si así se desea.

Una vez escogidos los componentes de la edición, se realizó un diseño que cumpliera con la funcionalidad y estética requerida tras el cual, al ser aprobado por el grupo de trabajo, se desarrolló la pantalla y cada uno de los *pop-ups* que ésta contiene, con las herramientas de diseño. En este paso, se realizó el controlador de eventos que permitiría manejar la extracción de cada uno de los elementos de la partitura y su renderizado, dando paso a otro período de pruebas sin la integración al *back end*, validando que tanto el encabezado como los elementos de la partitura fueran obtenidos de manera correcta. Adicionalmente, apoyando a la facilidad de uso de la plataforma, se implementaron botones de ayuda a lo largo de las pantallas que tienen interacción con el usuario. Éstos despliegan *pop-ups* con tutoriales de cómo usar cada función, como la creación de proyectos, la interacción con los archivos multimedia, y las distintas opciones de edición.

Luego se realizó el *ABCHandler*, controlador clave que permitiría manejar la edición de la partitura en la notación ABC y las validaciones musicales. Se investigó acerca de las reglas musicales más importantes al momento de escribir una partitura para desarrollar esta librería. Al comprenderlas, se implementaron los diferentes componentes de la edición, validando siempre las reglas y adaptando el estado de la partitura para que estas normas no se quebrantaran. Una vez terminada, se realizaron pruebas sobre su funcionalidad, con tal de integrar al controlador responsable de la captura de eventos. Al terminar la edición, se realizó el proxy que permitiría consumir los servicios expuestos en el servidor, concluyendo con el periodo de pruebas que valida que cada aspecto de la aplicación cumpliera tanto con los requisitos funcionales especificados, como con los aspectos estéticos y de interfaz requeridos.

Desarrollo del *back end*. El desarrollo del *back end* se divide en tres módulos principales: la implementación del modelo de aprendizaje de máquina, la transcripción musical, y el desarrollo del servidor de la aplicación.

Implementación del modelo de aprendizaje de máquina. El aprendizaje de máquina tiende a abordar tres conceptos principales: la detección de objetos, el cual permite saber si un conjunto de características está presente en los datos que se van a analizar; la segmentación semántica, la cual define una operación binaria que permite extraer las características presentes en los datos analizados –de manera que se separe a los elementos que pertenecen al conjunto de características de los que no– y la segmentación de instancias,

normalmente presentada como una extensión de la segmentación semántica, la cual permite extraer las características presentes y clasificarlas, obteniendo una diferenciación sobre las características extraídas (Ronneberger et al., 2015).

De acuerdo con las limitaciones y al alcance del proyecto, la solución se desarrolló bajo la segmentación semántica, con el objetivo de extraer los diferentes instrumentos presentes en la canción. El modelo empleado para realizar este tipo de separación fue *U-Net*, el cual fue diseñado para realizar separación semántica sobre conjuntos de imágenes biomédicas con pocos datos de entrada con los cuales entrenar al modelo, por lo que depende en gran medida del “*data augmentation*” (Jansson et al., 2017). Este modelo usa la arquitectura de redes completamente convolucionales, modificadas de tal manera que el modelo trabaje con muy pocas imágenes de entrenamiento y produzca segmentaciones más precisas.

La implementación puntual de *Harmonics* se basó en la adaptación del modelo *U-Net* que emplea espectrogramas para segmentar las voces humanas dentro de canciones. Como este modelo de segmentación requiere de un conjunto de datos con las imágenes originales y sus correspondientes segmentaciones para su entrenamiento (Jansson et al., 2017), *Harmonics* requiere de muestras de canciones completas, junto a la muestra de cada instrumento tocando su parte por separado. Este es el motivo por el que *Harmonics* tan solo puede trabajar con los instrumentos mencionados anteriormente.

Proceso de entrenamiento. Se utilizó la base de datos pública MUSDB18 (Rafii et al., 2017), la cual contiene el estándar de facto para la separación de fuentes. Esta base de datos contiene canciones de varios géneros musicales como pop, jazz, y rock. Actualmente cuenta con dos esquemas o alineaciones musicales: voz-acompañamiento y voz-batería-bajo-otros.

Las canciones son segmentadas, donde se extrae el espectrograma de cada uno, para procesarlos en bloques de 512×4096 , con saltos de 1024. El resultado final de la separación fue alcanzado en tres iteraciones. Dado que el entrenamiento del modelo depende en gran medida del proceso de *data augmentation*, se probaron las siguientes combinaciones de transformaciones y se encontraron los siguientes resultados.

En la primera iteración se utilizó la siguiente combinación:

- Transformación temporal: disminuye o aumenta la velocidad de la señal de acuerdo con un parámetro aleatorio
- Transformación temporal: disminuye la velocidad de la señal de acuerdo con un parámetro ya establecido en 25% de disminución.
- Transformación de afinación: aumenta, o disminuye, la frecuencia de la señal de acuerdo con un parámetro aleatorio, alterando el tono de la melodía.
- Transformación de afinación: aumenta la frecuencia de la señal de acuerdo con un parámetro ya establecido, alterando el tono de la melodía en una octava.
- Transformación de efecto de sonido: Se aplicaba el efecto Tremolo, con una reverberación de 0.5 y una oscilación establecida en 2.
- Transformación de efecto de sonido: Se aplicaba el efecto de sonido *Overdrive*, con una ganancia establecida en 4.

Los resultados de la separación de este intento presentaban un ruido considerable en cada pista, causado por los demás instrumentos, ya que los efectos de sonido seleccionados para trabajar con las distorsiones no trataban los tonos acústicos como se esperaba.

En la segunda iteración se utilizaron las mismas transformaciones, reemplazando los efectos de sonido usados por los siguientes:

- Transformación de efecto de sonido: Se aplicaba el efecto de sonido *Compressor*, establecido con una compresión de 0.5 para resaltar las frecuencias altas de la señal.
- Transformación de efecto de sonido: Se aplicaba el efecto de sonido de vibrato, con un 0.2 de variación en la frecuencia de la señal.

Los resultados de este intento fueron mucho más prometedores, tanto con canciones con tonos acústicos, como canciones con cierto grado de distorsión (como el rock). Aunque las predicciones del modelo aún separaban las voces de la canción con mucho ruido por los otros instrumentos presentes.

En la iteración final, solo se utilizaron las cuatro transformaciones básicas, de tiempo y de afinación de señal. Las predicciones del modelo mejoraron considerablemente al hacer uso de solo estos filtros, produciendo así una extracción de características mucho más fiable para el programa.

Proceso de separación. El controlador de separación itera sobre la canción de entrada en bloques de treinta segundos. Cada uno de estos es procesado por el modelo, obteniendo así las diferentes fuentes especificadas. Luego, se une cada una de las partes de la fuente extraída de la canción para generar un archivo .MP3. Así, se ejecuta el proceso de transcripción por cada una de las voces separadas, generando los diferentes archivos multimedia que contienen las transcripciones de las voces.

La transcripción musical de las voces separadas se implementó utilizando el algoritmo MELODIA, el cual combina el uso de espectros, suma de armónicos y cálculo de características para la extracción de afinaciones, convirtiéndolo en un algoritmo bastante confiable comparado con otros que solo emplean alguno de estos métodos para generar sus resultados (Salamon & Gomez, 2012).

Para generar las diferentes transcripciones, se decidió usar los archivos MIDI como interfaz unificada, dado que son un formato más estandarizado actualmente soportado por múltiples aplicaciones como *Musescore* o *FluidSynth*, entre otras. La generación del archivo MIDI representativo de la voz particular se genera a partir de las notas y duraciones generadas por MELODIA, discriminando todas las notas detectadas con una duración menor a 0.1 segundos, ya que estas notas describen generalmente ruido dentro del contexto de la melodía.

Una vez creado el MIDI, este debe ser normalizado. Este estado se consigue aproximando las duraciones de las notas inscritas a la subdivisión en la potencia de dos ($2^n/2^k$) más cercana. Es así como se provee el MIDI normalizado para obtener la renderización de la partitura en PDF, la transcripción ABC y la sinterización del audio, tras realizar el proceso de transcripción.

Desarrollo del servidor. La implementación del servidor de la aplicación se hizo utilizando *Django Rest Framework* (Encode OSS, 2021). Éste se desarrolló de manera modular de tal forma que se facilitara la integración entre los dos componentes descritos anteriormente. Este apartado se encarga de gestionar la lógica de negocio y las clases que permiten realizar las diferentes operaciones que comprende el sistema, además de los permisos de acceso para realizar acciones de edición y modificación sobre los proyectos almacenados, junto con el *API Rest* que el cliente de la aplicación utiliza para acceder a los recursos almacenados por el servidor.

2.4 Fase de pruebas

En esta fase se plantea cómo verificar el funcionamiento en general de *Harmonics* y la veracidad de las partituras producidas.

Prueba UAT. Para verificar si el proyecto realmente es viable en un mercado y confirmar su utilidad frente a la problemática propuesta, se realizaron pruebas por medio de encuestas a personas ajenas al equipo de desarrollo. El objetivo de estas pruebas era calificar la plataforma en términos de facilidad de uso, la utilidad y fidelidad de los resultados de los procesos de separación de fuentes, y la transcripción musical y la aceptación de diseño de interfaces gráficas.

Para dichas pruebas se buscaron personas que contaran con un conocimiento musical moderado, con tal que fueran capaces de validar algunas funciones de edición y tener criterio crítico sobre los resultados. Se logró encuestar a siete estudiantes de la carrera de música de la propia universidad, próximos a terminar su carrera. La encuesta contaba con catorce preguntas básicas, centradas en la facilidad de uso, el atractivo gráfico de la herramienta, y la fidelidad de los resultados.

Pruebas de comparación de partituras producidas. Para verificar el porcentaje de exactitud de las partituras producidas, se desarrolló un algoritmo que compara dos archivos MIDI –el esperado y el generado–. Este algoritmo trabaja bajo dos supuestos: la melodía extraída siempre intentará acercarse lo más posible a la melodía original y, si se compara el archivo MIDI producido con el archivo MIDI esperado por secciones, se reduciría el error de la comparación, ya que se limitaría la cantidad de valores extremos en el segmento comparado.

Para realizar esta comparación, los archivos MIDI se partitionaron en trozos de diez segundos, sumando todas las notas y, de este modo, comparar el valor esperado con el valor generado, calculando así los porcentajes de acierto. Como conjunto de datos esperados se utilizaron canciones de rock y rock progresivo que contaran con una transcripción a MIDI con múltiples voces.

3 Resultados

El desarrollo dió como fruto una aplicación web de diseño pulido y grandes funcionalidades para la pedagogía musical. Ésta se compone de ocho pantallas esenciales para la exposición de la funcionalidad: Inicio (una breve descripción del proyecto, junto a los diferentes botones para la navegabilidad entre las otras pantallas), Registro (donde un usuario podrá crear una cuenta en el sistema), Inicio de sesión (le permite al usuario acceder a su perfil en el sistema), Empezar (donde el usuario ingresará los datos para realizar la separación y la generación de partituras), Mis proyectos (donde el usuario podrá observar sus proyectos), Proyectos, (donde el usuario podrá ver todos los proyectos realizados por la comunidad tras filtrar su búsqueda), Visualizador de las partituras y los audios separados (donde el usuario podrá visibilizar su partitura y escuchar el audio separado), y el Editor de partituras (donde el usuario podrá editar sus partituras y escuchar su edición en vivo). La [Figura 2](#) presenta una de las pantallas de la aplicación, mientras que todas las pantallas pueden visualizarse en el video demostrativo disponible en <https://bit.ly/H4rm1kz>.



Figura 2. Ejemplo de la pista de voz de una canción procesada por *Harmonics*

3.1 Pruebas de UAT

Se seleccionaron preguntas ejemplares del modo en que los encuestados perciben el desempeño de los procesos de separación de fuentes y de transcripción musical. Adicionalmente, se consideró pertinente analizar la última pregunta de la encuesta, referente a la utilidad de la plataforma para el público objetivo.

Se observa como los resultados obtenidos presentan una moda de 7 sobre 10 dentro de la escala de satisfacción, en cada caso entre la transcripción musical, la edición de las partituras, y la separación del audio. Estos resultados se interpretan como positivos porque, no solo se encuentran en el percentil mayor de los resultados posibles ofrecidos a los usuarios, sino que, al comparar con las opiniones recolectadas de

las preguntas de respuesta abierta, se evidenció una aceptación general de la aplicación y sus funcionalidades.

Los usuarios también recomiendan las partituras generadas en la plataforma tanto a usuarios primerizos como a gente con experiencia en el campo de la música. Esto indica que los sujetos encuestados, bajo el criterio que han desarrollado a lo largo de su formación profesional, corroboran el análisis anterior al recomendar la plataforma como método de aprendizaje útil en general.

3.2 Comparación de partituras producidas

Los resultados obtenidos revelan que la partitura de una melodía compuesta de notas simples (sin acompañamiento) es semejante en un 68%, en promedio, con las partituras esperadas para la voz (ver [Figura 3](#)). En contraste, una melodía descrita por acompañamiento o partes melódicas, como en el caso de un piano, genera una partitura semejante en un 39% en promedio. Esto se debe, principalmente, a que el algoritmo usado, MELODIA, está diseñado de forma que funcione por medio de armónicos que le permiten identificar la onda fundamental, lo cual implica que le es imposible detectar o identificar acordes en la voz extraída.

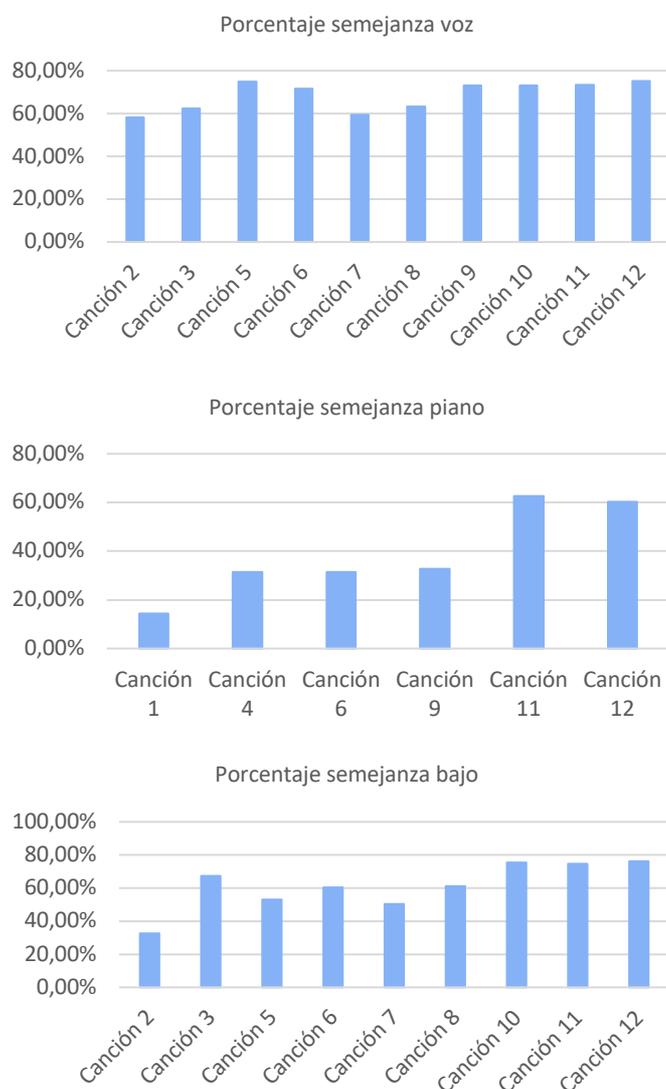


Figura 3. Porcentajes de semejanza en las partituras producidas

A continuación, se presenta la comparación entre dos segmentos de 10 segundos de los archivos MIDI para el bajo de la interpretación de la canción “*Stand by me*” de Ben E. King (ver Figura 4). El algoritmo toma estos segmentos y compara los valores numéricos de cada nota, para hacer una totalización de los valores para cada segmento. Esto es posible gracias a que los archivos MIDI representan las notas de forma numérica.



Figura 4. Comparación de segmentos de partitura para el instrumento Bajo.

Lo anterior es indicativo de una alta dependencia entre la calidad de las partituras producidas y el tipo de instrumento que se seleccione. Nótese como, en el caso de la voz, las peores medidas se acercan a un 60%. Ha de hacerse énfasis en cómo en el algoritmo de prueba desarrollado para *Harmonics*, las notas de un segmento de canción para un instrumento dado se representan con un valor numérico, el cual se compara con el resultado del mismo proceso para el mismo segmento generado por *Harmonics*. En suma, entre más simple es la melodía separada, mejor es el resultado que *Harmonics* puede generar.

4 Conclusiones y trabajo futuro

Este proyecto apuntaba a la generación de partituras por medio de la transcripción musical, con el objetivo de incentivar y facilitar el aprendizaje musical para novicios. No solo se evidencia su cometido funcional, sino que se consiguió ofrecer una herramienta útil a aquellos experimentados, como se evidenció en las encuestas.

A raíz de los resultados obtenidos en el primer acercamiento a la separación de fuentes con los algoritmos de BSS, puede afirmarse con toda certeza que éste no es un método viable para alcanzar el objetivo planteado, ya que estos algoritmos están restringidos a señales con múltiples canales de transmisión. En contraste, la tecnología aplicada posteriormente constituye una aproximación bastante eficaz al ser capaz de extraer las voces de los diferentes instrumentos en una única canción de entrada, lo que facilitó evolucionar el proceso, mejorando la extracción de características con el simple hecho de encontrar más datos con los cuales alimentar al modelo encargado de la separación.

El modelo de aprendizaje de máquina utilizado, junto a la implementación del algoritmo MELODIA, resultó ser bastante flexible, ya que depende del conjunto de datos empleados para entrenar, por la mayor parte. En contraste, la investigación sobre BSS por su cuenta tan solo reveló varias herramientas obsoletas, incompatibles o insuficientes para el problema como se había planteado. Se concluye que una aproximación desde el aprendizaje de máquina es la más apropiada para la situación.

Por otro lado, se evidencia un gran avance en la identificación, extracción y transcripción de melodías por medio del algoritmo MELODIA, aunque no son perfectamente usables todavía. Al concluir el proceso de pruebas, comparando las melodías generadas con las melodías esperadas y su respectiva validación con expertos, se observaron resultados de hasta 76% de semejanza. Por esto se concluye que el uso de MELODIA como el algoritmo transcriptor constituye una primera aproximación adecuada.

Con base en los resultados y las restricciones del proyecto, se concluye que éste presenta grandes oportunidades de mejora. La más evidente estaría dada por el número de instrumentos que puede soportar la separación de fuentes: actualmente, se encuentra limitado por los datos que alimentan a la red neuronal. Al ingresar nuevos instrumentos con *samples* apropiados, se lograría que la red pudiera reconocer y separar nuevos instrumentos como guitarras, charangos, quenás, entre otros. Además, al hablar de percusiones, se podría crear un nuevo modelo que pudiera diferenciar cada uno de los elementos presentes en la separación realizada y así, ahondar en el desarrollo de partituras de batería.

Otra mejora evidente es la corrección del error que está disminuyendo la calidad del audio producido. Es probable que resolver este problema consiga que las personas interesadas, como aquellos encuestados, consideren que *Harmonics* es de aún mayor calidad y así, se podría convencer a los posibles clientes de invitar a más personas, formando un mercado donde innovar.

Sobre la transcripción musical podría implementarse otra mejora: actualmente, el algoritmo se enfoca en obtener la nota dominante (la más baja de la onda), lo que causa que la transcripción presente problemas con pianos a dos manos o con acordes. Por lo tanto, al optimizar el algoritmo de modo que sea capaz de detectar múltiples notas en una frecuencia, se podrían revelar acordes y melodías además de aquellas basadas en la frecuencia fundamental del audio.

Finalmente, se podría pensar en incluir las mismas funcionalidades que *ABCjs* soporta en el editor de texto. Se requerirían actualizar la pantalla de la edición y el controlador que dinamiza la página, además de extraer cada uno de los elementos de la notación y la librería propia *ABCHandler*, procurando cumplir con las normativas musicales.

Declaración de conflicto de intereses

Los autores declaran no tener conflicto de intereses con respecto a la investigación, autoría o publicación de este artículo.

Financiación

Los autores no recibieron apoyo financiero para la investigación, autoría y/o publicación de este artículo.

ORCID iD

Randy Darrell Lancheros-Molano  <https://orcid.org/0000-0001-6919-8930>

Juan Sebastián Triana-Perez  <https://orcid.org/0000-0003-4903-3631>

Juan Felipe Castañeda-Chaparro  <https://orcid.org/0000-0002-0910-4311>

Felipe Andrés Gutiérrez-Naranjo  <https://orcid.org/0000-0002-4088-5590>

Andrea del Pilar Rueda-Olarte  <https://orcid.org/0000-0002-9245-7328>

Referencias

- Agile Alliance. (2021). *Kanban*. <https://www.agilealliance.org/glossary/kanban/>
- Byrne, R. W., & Russon, A. E. (1998). Learning by imitation: A hierarchical approach. *Behavioral and Brain Sciences*, 21(5), 667–684. <https://doi.org/10.1017/S0140525X98001745>
- Cano, E., FitzGerald, D., Liutkus, A., Plumbley, M. D., & Stoter, F.-R. (2019). Musical Source Separation: An Introduction. *IEEE Signal Processing Magazine*, 36(1), 31–40. <https://doi.org/10.1109/MSP.2018.2874719>
- Duan, Z., Mysore, G. J., & Smaragdis, P. (2012). Online PLCA for Real-Time Semi-supervised Source Separation. In F. Theis, A. Cichocki, A. Yeredor, & M. Zibulevsky (Eds.), *Latent Variable Analysis and Signal Separation. LVA/ICA 2012. Lecture Notes in Computer Science* (pp. 34–41). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-28551-6_5
- Encode OSS. (2021). *Django REST framework*. <https://www.django-rest-framework.org>
- Gao, B., Woo, W. L., & Dlay, S. S. (2008). Single channel audio source separation. *WSEAS Transactions on Signal Processing*, 4(4), 173–182.
- Gómez, E., Grachten, M., Hanjalic, A., Janer, J., Jordà, S., Julià, C. F., Liem, C., Martorell, A., Schedl, M., & Widmer, G. (2013). PHENICX: Performances as Highly Enriched and Interactive Concert Experiences. *SMAC Stockholm Music Acoustics Conference 2013 and SMC Sound and Music Computing Conference 2013*, 1–8.
- He, P., She, T., Li, W., & Yuan, W. (2018). Single channel blind source separation on the instantaneous mixed signal of multiple dynamic sources. *Mechanical Systems and Signal Processing*, 113, 22–35.

- <https://doi.org/10.1016/j.ymsp.2017.04.004>
- Hennequin, R., Khlif, A., Voituret, F., & Moussallam, M. (2020). Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5(50), 2154. <https://doi.org/10.21105/joss.02154>
- Jansson, A., Humphrey, E., Montecchio, N., Bittner, R., Kumar, A., & Weyde, T. (2017). Singing voice separation with deep U-Net convolutional networks. *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, 745–751.
- Jouny, I. (2007). Improving music algorithm using BSS. *2007 IEEE Antennas and Propagation Society International Symposium*, 5267–5270. <https://doi.org/10.1109/APS.2007.4396735>
- Lebler, D. (2008). Popular music pedagogy: peer learning in practice. *Music Education Research*, 10(2), 193–213. <https://doi.org/10.1080/14613800802079056>
- Naik, G. R., & Wang, W. (2014). *Blind Source Separation. Advances in Theory, Algorithms and Applications* (G. R. Naik & W. Wang (eds.)). Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-642-55016-4>
- Pepino, L., & Bender, L. (2018). Separación de fuentes musicales mediante redes neuronales convolucionales con múltiples decodificadores. *IV Jornadas JAAS 2018*, 1–7.
- Peretz, I. (2006). The nature of music from a biological perspective. *Cognition*, 100(1), 1–32. <https://doi.org/10.1016/j.cognition.2005.11.004>
- Rafii, Z., Liutkus, A., Stöter, F.-R., Mimitakis, S. I., & Bittner, R. (2017). *The MUSDB18 corpus for music separation*. <https://doi.org/10.5281/zenodo.1117372>
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science* (pp. 234–241). Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28
- Roulston, K., Jutras, P., & Kim, S. J. (2015). Adult perspectives of learning musical instruments. *International Journal of Music Education*, 33(3), 325–335. <https://doi.org/10.1177/0255761415584291>
- Salamon, J., & Gomez, E. (2012). Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6), 1759–1770. <https://doi.org/10.1109/TASL.2012.2188515>
- Shinde, P. P., & Shah, S. (2018). A Review of Machine Learning and Deep Learning Applications. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCCUBEA)*, 1–6. <https://doi.org/10.1109/ICCCUBEA.2018.8697857>
- Wytock, M., & Kolter, J. (2014). Contextually Supervised Source Separation with Application to Energy Disaggregation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 486–492. <https://ojs.aaai.org/index.php/AAAI/article/view/8769>
- Yu, X., Hu, D., & Xu, J. (2013). *Blind Source Separation: Theory and Applications*. John Wiley & Sons, Inc.