Qwen

Análisis Exhaustivo de Librerías y Herramientas para la Detección del Complejo QRS en Señales ECG

Evaluación Comparativa de Librerías de Python para la Detección de QRS

El ecosistema de Python ha madurado considerablemente en el ámbito del procesamiento de señales biomédicas, ofreciendo a investigadores y desarrolladores clínicos un conjunto diverso y potente de herramientas para el análisis de electrocardiogramas (ECG). La elección de la librería adecuada depende críticamente de los objetivos del proyecto, ya sea la maximización de la velocidad de procesamiento, la capacidad de realizar comparaciones algorítmicas exhaustivas, la integración en flujos de trabajo de ciencia de datos existentes, o la necesidad de delineación avanzada de ondas. Las librerías disponibles se pueden clasificar en varias categorías principales, cada una con sus propias fortalezas y áreas de aplicación ideales.

Una categoría fundamental está compuesta por librerías que implementan algoritmos clásicos y bien documentados, proporcionando un control granular sobre el proceso de detección. Entre ellas, `py-ecg-detectors` destaca por su enfoque exhaustivo, ofreciendo acceso a ocho algoritmos distintos para la detección de latidos, incluyendo Pan-Tompkins, Hamilton, Christov, Engelse y Zeelenberg, Stationary Wavelet Transform (SWT), Two Moving Average, Matched Filter y WQRS [5] [8]. Esta característica única permite a los investigadores comparar directamente el rendimiento de diferentes métodos en el mismo conjunto de datos, una capacidad invaluable para la validación y la selección de algoritmos. La interfaz unificada, accesible a través de una clase `Detectors` inicializada con la frecuencia de muestreo, simplifica enormemente la experimentación [5]. Además, el paquete incluye un módulo `hrv` que realiza un análisis completo de la variabilidad de la frecuencia cardíaca (HRV) a partir de los intervalos RR, creando un flujo de trabajo integral dentro de un único paquete de software [5].

Otra librería notable en esta categoría es `wfdb-python`, que va mucho más allá de una simple función de detección. Forma parte de un ecosistema completo diseñado para acceder, procesar y analizar bases de datos de PhysioNet, lo que la convierte en una herramienta indispensable para la investigación académica y clínica [3]. Su detector XQRS es uno de los más citados y evaluados en la literatura, demostrando una alta precisión, con un 99.81% de sensibilidad (recall) y un 99.58% de precisión (positive predictivity) en el desafiante MIT-BIH Polysomnographic Database [1]. Un aspecto crucial de `wfdb-python` es su integración con otras herramientas del WFDB, como `correct_peaks`, que permite refinar la posición de los picos detectados para mejorar aún más la precisión temporal, compensando los retrasos introducidos por los filtros internos del algoritmo [3] [7]. Esta funcionalidad de post-procesamiento es vital para aplicaciones que requieren una localización precisa de los eventos cardíacos.

En contraste con las librerías enfocadas en la comparación algorítmica, existe un grupo de soluciones optimizadas para el rendimiento, particularmente en velocidad. `sleepecg` es el ejemplo paradigmático de esta categoría; implementa una versión altamente optimizada en lenguaje C del algoritmo de Pan-Tompkins, lo que le confiere una velocidad de procesamiento excepcional [1]. Según benchmarks, es capaz de procesar aproximadamente 20 horas de señal ECG por segundo en un portátil de gama media, manteniendo al mismo tiempo un alto nivel de rendimiento en términos de precisión [1]. Para proyectos que implican sistemas de monitoreo en tiempo real, grandes volúmenes de datos o la necesidad de procesamiento rápido en dispositivos embebidos, `sleepecg` representa una opción tecnológicamente superior. Por otro lado, `fast_qrs_detector`, una librería desarrollada recientemente por Aura Healthcare, se posiciona como una alternativa moderna que, según sus propios benchmarks internos, supera a las librerías públicas existentes tanto en precisión como en eficiencia, siendo en promedio cinco veces más rápida que el mejor algoritmo en su comparación [2]. Este desarrollo se basa en una aproximación híbrida que combina ideas de múltiples métodos publicados recientes, lo que sugiere un camino hacia futuras innovaciones en el campo [2].

La siguiente tabla resume las características clave de las librerías de Python más prominentes mencionadas en las fuentes:

Librería	Algoritmos Principales	Fortalezas Clave	Módulos Adicionales	Integración
neurokit2	No especificado [4]	Flujo de trabajo integrado end-to-end, delineación avanzada de ondas (P, Q, S, T) con múltiples métodos (CWT, DWT) [4]	Integración nativa con pandas y numpy [4], plotting integrado [4]	Integración con bibliotecas de ciencia de datos estándar (numpy, pandas) [4]
wfdb-python	GQRS, XQRS [3] [7]	Alta precisión, robustez, integración completa con bases de datos de PhysioNet, post-corrección de picos (`correct_peaks`) [1] [3]	Lectura y escritura de archivos WFDB, análisis de anotaciones [3]	Parte del ecosistema WFDB para acceso a bases de datos de PhysioNet [3]
py-ecg-detectors	Pan-Tompkins, Hamilton, Christov, SWTS, etc. (8 total) [5] [8]	Capacidad de comparar múltiples algoritmos en el mismo conjunto de datos, interfaz unificada [5]	Módulo `hrv` para análisis de HRV (RMSSD, LF/HF ratio, etc.) [5]	Interfaz de usuario simple basada en una clase `Detectors[[5]]`
biosppy	Información no disponible en las fuentes proporcionadas	Fiable, establecida en la comunidad de investigación, centrada en la detección de R-peaks [6] [25]	Funciones de clustering y biometría [25]	Biblioteca generalista para el procesamiento de señales biomédicas [25]
sleepecg	Pan & Tompkins (implementación en C) [1]	Velocidad de procesamiento extremadamente alta (20h/s en laptop) [1]	Información no disponible en las fuentes proporcionadas	Optimizado en C para máxima eficiencia [1]
fast_qrs_detector	Híbrido (basado en Zidelmal, Lu, Modak, Šarlija) [2]	Alta velocidad (promedio 5x más rápido que librerías públicas) y precisión comparable/better [2]	Visualización básica de resultados (`print_signal_with_qrs`) [2]	Instalable vía pip con dependencias estándar (numpy, scipy, pandas) [2]

Un tercer grupo de librerías busca abstraer la complejidad del procesamiento de señales, ofreciendo funciones de alto nivel que realizan todo el proceso de principio a fin. En esta categoría, `neurokit2` se erige como una de las opciones más completas. Proporciona una función `ecg_process()` que encapsula el preprocesamiento, la detección de R-picos (`ecg_peaks()`) y la delineación de todas las ondas cardíacas [4]. Su método de delineación, `ecg_delineate()`, es particularmente potente, ya que implementa tres técnicas distintas: 'peak' (búsqueda de picos individuales), 'cwt' (Transformada de Ondalet Continua) y 'dwt' (Transformada de Ondalet Discreta), siendo esta última el método predeterminado [4]. La capacidad de comparar visual y cuantitativamente los resultados de estos diferentes métodos es una característica inmensamente valiosa para la investigación clínica, permitiendo seleccionar el método óptimo según la calidad de la señal o el objetivo específico del estudio. Por ejemplo, se ha observado que el método DWT es superior para la delineación de la onda P, mientras que el método CWT proporciona mejores resultados para determinar los bordes (onset/offset) de

las ondas P y T [4]. La integración nativa de `neurokit2` con las bibliotecas de ciencia de datos de Python como pandas y numpy facilita su uso en entornos de análisis de datos estándar, permitiendo una fácil manipulación y visualización de los resultados [4].

Finalmente, existen soluciones más especializadas o emergentes. `heartpy`, aunque primordialmente diseñada para señales de fotoplectismografía (PPG), fue incluida en un benchmark comparativo donde mostró un rendimiento inferior en la detección de QRS en señales ECG debido a un mayor número de falsos positivos [1]. Esto subraya la importancia de elegir herramientas diseñadas específicamente para la modalidad de señal en cuestión. Por otro lado, `bioSigKit` es una herramienta que implementa el algoritmo Pan-Tompkins paso a paso, sirviendo como un excelente recurso educativo y para la investigación que requiere un control detallado sobre cada etapa del procesamiento de la señal [9]. Aunque se presenta como un toolbox para investigación y no para diagnóstico médico, su claridad pedagógica es innegable [9].

Para un investigador o desarrollador clínico, la elección de la librería óptima no es una decisión binaria. Depende del equilibrio deseado entre simplicidad, control, velocidad y funcionalidad avanzada. Para un inicio rápido y un flujo de trabajo completo que incluya la delineación de ondas, `neurokit2` es probablemente la opción más versátil y poderosa [4]. Si el proyecto requiere la máxima velocidad de procesamiento, como en sistemas de monitoreo continuo, `sleepecg` es un candidato ideal debido a su implementación en C [1]. Y si el trabajo se centra en bases de datos estándar de PhysioNet y requiere la máxima robustez y adherencia a los estándares de la comunidad de investigación, `wfdb-python` es la referencia indiscutible [3]. Para la investigación comparativa que busca explorar el espacio de los algoritmos, `py-ecg-detectors` es la herramienta perfecta [5]. La elección estratégica de la librería correcta es el primer paso fundamental para garantizar el éxito de cualquier proyecto de análisis de ECG.

Fundamentos Algorítmicos: El Paradigma Clásico de Pan-Tompkins y sus Variantes

En el vasto campo del procesamiento de señales ECG, un algoritmo domina históricamente y sigue siendo relevante en la actualidad: el algoritmo de Pan-

Tompkins, originalmente publicado en 1985 [9] [15]. Este método no es simplemente un hito histórico; constituye la piedra angular sobre la cual se han construido muchas de las soluciones modernas, tanto en librerías de Python como en software comercial. Comprender su mecanismo central es fundamental para cualquier investigador o desarrollador clínico, ya que proporciona el lenguaje común para diagnosticar problemas, interpretar resultados y evaluar críticamente el rendimiento de cualquier herramienta que lo utilice. Su persistencia radica en una combinación de robustez, fiabilidad y una eficiencia computacional notable, especialmente cuando se implementa de manera optimizada.

El núcleo del algoritmo de Pan-Tompkins reside en una cascada de cuatro etapas de procesamiento digital cuidadosamente diseñadas para transformar la señal ECG en un formato donde el complejo QRS es fácilmente identificable [15] [30]. La primera etapa es un filtro pasa-banda, típicamente con un rango de 5 a 15 Hz [9] [30]. Este filtro tiene dos funciones cruciales: primero, suprime el ruido de baja frecuencia como la deriva del punto de base (baseline wander) y el ruido de línea (powerline noise); segundo, resalta la energía contenida en el complejo QRS, que ocupa principalmente esta banda de frecuencias, mientras atenúa las ondas P y T, que tienen componentes de frecuencia más bajas y más altas, respectivamente [30]. La segunda etapa es un filtro diferenciador, a menudo implementado como un filtro de diferencia de cinco puntos [15] [30]. Este operador resalta los cambios abruptos en la amplitud de la señal, que corresponden a los bordes ascendentes y descendentes de la onda R. La tercera etapa consiste en elevar al cuadrado la salida del diferenciador. Esta operación no lineal amplifica aún más las frecuencias más altas, haciendo que el pico del complejo QRS se vuelva aún más prominente, mientras que el ruido residual se mitiga [15]. Finalmente, la cuarta etapa es un integrador de ventana móvil (moving-window integrator), que promedia la señal durante un período corto, típicamente de 150 ms (o 30 muestras a una tasa de 200 Hz) [9] [15]. Este paso estabiliza la amplitud del pico del QRS y reduce la sensibilidad al ruido, resultando en una señal de salida donde cada complejo QRS aparece como un pico bien definido y robusto.

Más allá de la transformación de la señal, el algoritmo de Pan-Tompkins incorpora sofisticados mecanismos de detección y post-procesamiento. Utiliza umbrales adaptativos dinámicos, un concepto clave para su robustez en condiciones de señal variables. Estos umbrales se calculan en tiempo real utilizando estimaciones corrientes de los picos de QRS (SPKI/SPKF) y los picos de ruido (NPKI/NPKF) [15]. Después de detectar un complejo QRS, el umbral de detección se reduce para facilitar la detección de latidos posteriores. Para evitar falsos positivos causados por

ondas T altas, el algoritmo implementa un período refractario (refractory period), durante el cual no se aceptan nuevos picos de detección [15] [34]. Si no se detecta un nuevo complejo QRS dentro de un tiempo esperado (por ejemplo, 1.66 veces el intervalo RR reciente), el algoritmo activa un mecanismo de búsqueda atrás (search-back) con un umbral más bajo para recuperar posibles latidos perdidos [15] [34]. Estos mecanismos, junto con la estimación del intervalo RR medio de los últimos ocho latidos para ajustar el comportamiento del detector, hacen que el algoritmo sea extremadamente resistente a las variaciones de la frecuencia cardíaca y al ruido ambiental [22].

La popularidad y el éxito del algoritmo de Pan-Tompkins han llevado a su implementación y optimización en una multitud de plataformas y lenguajes de programación. En el mundo de Python, la librería `sleepecg` ofrece una implementación altamente optimizada en C, que aprovecha la velocidad de un lenguaje compilado para lograr tiempos de procesamiento excepcionales, siendo capaz de manejar decenas de horas de ECG por segundo en hardware de consumo [1]. De manera similar, la librería `wfdb-python` también utiliza variantes del algoritmo PT en sus propios detectors, como GQRS y XQRS, integrándolos en un ecosistema más amplio para el análisis de bases de datos fisiológicas [3] [7]. El algoritmo también ha sido objeto de numerosas variantes y optimizaciones para su implementación en sistemas embebidos y microcontroladores. Por ejemplo, versiones en ANSI C han sido desarrolladas para monitores cardíacos CE-marked, donde se ha demostrado una sensibilidad de $\geqslant$99.74% y una positividad de $\geqslant$99.79% en bases de datos estándar como MIT/BIH [22]. Estas implementaciones a menudo simplifican ciertos aspectos del algoritmo original para reducir la carga computacional, como utilizar medias en lugar de medianas para la estimación del intervalo RR o reducir la longitud del período refractario [22]. Incluso se han creado versiones simplificadas para plataformas de bajo costo como Arduino, basadas en la idea de un sistema de promedio móvil para filtrado, adaptando los principios fundamentales del algoritmo a recursos limitados [29].

La influencia del algoritmo Pan-Tompkins trasciende las implementaciones puramente digitales. El propio diseño del algoritmo ha inspirado nuevas aproximaciones híbridas. La librería `fast_qrs_detector`, por ejemplo, dibuja su inspiración conceptual de múltiples métodos publicados, incluyendo la utilización de coeficientes de ondaletas (inspirado en Zidelmal et al.), umbralización adaptativa mejorada (Lu et al.), y técnicas de umbralización multinivel (Modak et al.), demostrando cómo los principios de robustez y adaptación del PT siguen siendo relevantes en el diseño de nuevos algoritmos [2]. De hecho, muchos de los

benchmarks de rendimiento de nuevos algoritmos se comparan con el rendimiento de las implementaciones de Pan-Tompkins como un punto de referencia estándar [11] [35]. La evaluación de la performance de un nuevo detector a menudo implica calcular métricas como la sensibilidad (señalización de todos los latidos verdaderos) y el valor predictivo positivo (evitar falsos positivos), y el algoritmo Pan-Tompkins ha alcanzado niveles de rendimiento muy altos en estas métricas, con estudios reportando una exactitud de detección del 99.3% en el MIT/BIH arrhythmia database [15].

Para el investigador o desarrollador clínico, la familiaridad con el algoritmo Pan-Tompkins es, por tanto, una competencia crítica. Permite ir más allá de la caja negra de una librería y entender cómo y por qué funciona. Cuando se observan resultados inesperados, como latidos perdidos en un segmento ruidoso o falsos positivos en presencia de ondas T prominentes, el conocimiento de los principios del PT (como la adaptación de umbrales y el período refractario) permite un diagnóstico más profundo del problema. Además, al leer la literatura científica, es común encontrar referencias a "el detector de Pan-Tompkins" o "la implementación de Pan-Tompkins", lo que significa que tener una comprensión sólida de este algoritmo es esencial para la participación efectiva en la conversación técnica y científica en este dominio. Es la base común sobre la cual se construyen las innovaciones y las discusiones sobre el estado del arte en la detección de QRS.

La Revolución de la Inteligencia Artificial en la Delineación de Ondas ECG

Si bien los algoritmos clásicos como Pan-Tompkins han dominado la detección robusta de R-picos, el análisis de señales ECG de alto valor clínico a menudo requiere una tarea mucho más compleja: la delineación precisa de todas las ondas cardíacas (P, Q, S, T) y sus bordes (inicios y finales). La localización exacta de estos puntos fiduciales es crucial para calcular mediciones como la duración del QRS (QRSd), la prolongación del segmento ST, la morfología de la onda T, y otros parámetros que son indicadores importantes de diversas patologías cardíacas [32] [33]. Aquí es donde el paradigma de la inteligencia artificial, y en particular las redes neuronales, está comenzando a tener un impacto revolucionario, ofreciendo una precisión que supera con creces a las metodologías tradicionales.

El ejemplo más paradigmático de esta nueva frontera es BRAVEHEART, un software de código abierto desarrollado en MATLAB que utiliza una red neuronal de memoria a corto/largo plazo (LSTM) bidireccional para la anotación de puntos fiduciales [18] [19]. A diferencia de los métodos basados en wavelets o en la búsqueda de picos locales, que a menudo luchan con la ambigüedad en los bordes de las ondas, la LSTM de BRAVEHEART aprende a reconocer la forma completa del complejo cardíaco. El modelo se entrena con datos etiquetados por expertos cardiólogos, permitiéndole capturar patrones sutiles y contextuales que son difíciles de codificar manualmente. Los resultados obtenidos son extraordinarios: en dos conjuntos de pruebas independientes, la precisión del modelo alcanzó una precisión clínicamente validada, con errores medios inferiores a ±2 milisegundos para los puntos Q-onset (Qon), Q-offset (Qoff) y T-offset (Toff) en comparación con las anotaciones de referencia [18]. Más del 99% de las anotaciones de Qon/Qoff y más del 98% de las anotaciones de Toff caen dentro de un margen de ±20 ms respecto a las anotaciones manuales de los expertos [18]. Esta capacidad de delineación de alta fidelidad abre la puerta a un nuevo nivel de análisis automatizado de ECG, donde las mediciones derivadas son mucho más fiables y reproducibles.

El avance de las redes neuronales no se limita a la delineación. También están mejorando continuamente el estado del arte en la detección de QRS. Un estudio propuso un detector de QRS completamente convolucional (FCN) que utiliza una arquitectura de Red de Pirámides de Características (FPN) como backbone, junto con convoluciones en profundidad para reducir el número de parámetros y optimizar el rendimiento [13]. Este modelo, entrenado en múltiples conjuntos de datos públicos, demuestra un rendimiento competitivo, destacando la flexibilidad de las arquitecturas de deep learning para adaptarse a una variedad de condiciones de señal. Otro enfoque interesante combina la Transformada de Ondalets Discretas (DWT) con una LSTM. En este sistema, la DWT se utiliza para la extracción de características de la señal ECG, que luego se alimentan a una red LSTM para la clasificación de arritmias [32]. Este enfoque híbrido aprovecha las fortalezas de ambos mundos: la DWT para el preprocesamiento y la extracción de características locales, y la LSTM para aprender patrones temporales complejos a nivel de ciclo cardíaco completo.

A pesar de su potencial, las redes neuronales presentan desafíos y consideraciones importantes. Primero, requieren grandes cantidades de datos de entrenamiento etiquetados, que son costosos y laboriosos de obtener mediante anotaciones manuales por parte de cardiólogos. Segundo, los modelos de deep learning son a menudo considerados "cajas negras", lo que dificulta la interpretación de cómo

llegan a sus decisiones, un obstáculo significativo en aplicaciones clínicas donde la transparencia es crucial. Tercero, el entrenamiento y la inferencia de estos modelos pueden ser computacionalmente intensivos, lo que podría representar un impedimento para su implementación en dispositivos embebidos de bajo consumo energético. Sin embargo, para aplicaciones donde la delineación exacta de las ondas es primordial —como en la investigación de síndromes de QT largo, la evaluación de la sincronización cardíaca en terapias CRT, o el análisis de la isquemia miocárdica—, los beneficios de la precisión de los modelos de IA superan con creces sus desventajas.

Para los desarrolladores clínicos y los investigadores que trabajan principalmente en Python, esto significa que deben estar al tanto de este cambio de paradigma. Mientras que librerías como `neurokit2` ya ofrecen métodos de delineación basados en wavelets (un método de IA más tradicional), que son buenos para una delineación de buena calidad, los modelos de deep learning representan la dirección futura para la máxima precisión [4]. Para implementar estas capacidades, se requerirán frameworks de aprendizaje automático como PyTorch o TensorFlow [13]. Una estrategia pragmática podría ser comenzar con una librería de Python como `neurokit2` para un análisis preliminar y rápido, y luego, para los casos de estudio o proyectos que requieran la máxima precisión, recurrir a un modelo de deep learning personalizado o a herramientas especializadas externas. La existencia de software como BRAVEHEART, aunque desarrollado en MATLAB, subraya la importancia de la interoperabilidad entre plataformas. Como se discutirá más adelante, la capacidad de invocar herramientas de MATLAB desde Python podría permitir a los equipos de desarrollo de Python aprovechar estas rutinas de cómputo de precisión sin abandonar su ecosistema de desarrollo preferido. La era de la delineación de ECG automatizada está evolucionando rápidamente, pasando de métodos basados en reglas a modelos de aprendizaje profundo que prometen un nivel de detalle y fiabilidad sin precedentes.

El Rol Estratégico de MATLAB: Herramientas Avanzadas e Interoperabilidad con Python

A pesar de la pregunta específica sobre Python, es imposible ignorar la influencia profunda y duradera de MATLAB en el campo del procesamiento de señales biomédicas, incluido el análisis de ECG. Muchas de las herramientas fundamentales, los estándares de validación y los algoritmos de referencia fueron desarrollados,

probados y perfeccionados en el ecosistema de MATLAB. Para un investigador o desarrollador clínico, pensar únicamente en términos de Python sería una limitación estratégica. MATLAB no solo es una plataforma de desarrollo en sí misma, sino que también actúa como un repositorio de herramientas de alto valor agregado y una puerta de entrada a un corpus de conocimiento científico que es a menudo indispensable para la validación rigurosa de los resultados.

El rol de MATLAB se manifiesta de varias maneras. Primero, es la plataforma de origen para muchas de las herramientas que luego se implementan en Python. El paquete de software WFDB (WaveForm DataBase), que es el estándar de facto para el acceso a bases de datos de PhysioNet, fue desarrollado originalmente en MATLAB [28] [35]. Sus componentes de línea de comandos, como `wqrs` (implementación del detector WQRS), `sqrs` (detector SQRS) y `gqrs` (detector GQRS), así como la herramienta `bxb` para el cálculo del error de detección, forman la columna vertebral de la mayoría de los benchmarks y estudios de validación de algoritmos de QRS [28] [35]. Por lo tanto, para validar científicamente cualquier resultado obtenido con una librería de Python, es a menudo necesario compararlo con los resultados generados por estas herramientas de referencia de MATLAB. Esta conexión hace que MATLAB sea una herramienta de validación crítica, incluso para aquellos que prefieren trabajar en Python.

Segundo, MATLAB alberga paquetes de software sofisticados y de código abierto que van mucho más allá de la simple detección de QRS, ofreciendo funcionalidades especializadas de alta precisión. BRAVEHEART es quizás el ejemplo más destacado, ya que, como se mencionó anteriormente, utiliza redes neuronales para la delineación de ondas con una precisión clínicamente validada que es difícil de igualar con métodos tradicionales [18] [19]. Otro ejemplo es HRVAS (Heart Rate Variability Analysis Software), un paquete dedicado exclusivamente al análisis exhaustivo de la variabilidad de la frecuencia cardíaca (HRV) [24]. Para la calificación de la calidad de la señal (SQA), `ecgScorer` es una toolbox de alta calidad que implementa un método novel para evaluar la usabilidad clínica de una señal, además de extraer hasta 37 índices de calidad de la señal (SQIs) [21]. Finalmente, BioSig es un toolkit materno y versátil para el procesamiento de diversas señales biomédicas, que ha influido en el desarrollo de muchas librerías de código abierto [25]. Estas herramientas representan décadas de investigación y desarrollo y ofrecen capacidades que pueden tardar mucho tiempo en replicarse en el ecosistema de Python.

La verdadera oportunidad estratégica, sin embargo, reside en la interoperabilidad. MATLAB ofrece una solución elegante y potente para que los usuarios de Python accedan a este vasto repositorio de herramientas de clase mundial. La solución es la MATLAB Engine API for Python, que permite iniciar una sesión de MATLAB directamente desde un script o notebook de Python[37][39][40][41]. Con esta API, un desarrollador de Python puede ejecutar casi cualquier comando o función de MATLAB, transferir datos (como arrays NumPy) entre los dos entornos, y recuperar los resultados generados por MATLAB para continuar con el análisis en Python[39]. Esta capacidad elimina la dicotomía de "Python versus MATLAB" y abre un espectro mucho más amplio de posibilidades técnicas.

La aplicación práctica de esta interoperabilidad es inmensa. Imagine un pipeline de análisis de ECG modulado para un investigador clínico: 1.Preprocesamiento y Detección en Python:El flujo de trabajo comienza en Python utilizando una librería como `neurokit2` o `wfdb-python` para cargar los datos, realizar un preprocesamiento inicial y detectar los R-picos de forma rápida y eficiente. 2.Validación en MATLAB:Para asegurar la máxima fiabilidad, los R-picos detectados se envían a MATLAB a través del engine API para ser verificados o refinados utilizando los algoritmos de referencia de WFDB, como `wqrs`. 3.Delineación de Ondas de Alta Precisión:Para obtener los bordes de las ondas P y T con la máxima precisión, el pipeline llama a una función de MATLAB que utiliza el modelo de BRAVEHEART para la delineación de alta fidelidad. 4.Análisis de HRV y Calidad de Señal:Los intervalos RR se envían a MATLAB para ser analizados con HRVAS, y la señal bruta se pasa a `ecgScorer` para una calificación exhaustiva de la calidad. 5.Análisis Final y Visualización en Python:Los resultados finales (coordenadas de las ondas, métricas HRV, puntuaciones de calidad) se devuelven a Python, donde se utilizan bibliotecas como Matplotlib y Pandas para generar informes, visualizaciones y análisis estadísticos finales.

Este enfoque híbrido permite al desarrollador clínico aprovechar lo mejor de ambos mundos: la flexibilidad, la riqueza del ecosistema de paquetes y la velocidad de desarrollo de Python, combinada con la robustez, la precisión y las herramientas de validación clínicamente validadas de MATLAB. Además, MATLAB también permite exportar modelos de aprendizaje automático a formatos compatibles con Python (como ONNX o TensorFlow), y viceversa, lo que facilita la integración de modelos de deep learning entrenados en un entorno con el otro[39][41]. La interoperabilidad bidireccional, que también incluye la capacidad de llamar a bibliotecas de Python desde MATLAB, crea un ecosistema colaborativo que acelera la investigación y el desarrollo de aplicaciones clínicas seguras y fiables[37][40]. Para el usuario, adoptar

una mentalidad híbrida es la estrategia más poderosa para navegar el complejo panorama de las herramientas de análisis de ECG.

Metodologías de Validación y Calidad de la Señal en el Procesamiento de ECG

La adopción de una librería de Python o MATLAB para el análisis de ECG no es una decisión que deba tomarse a ciegas. Para garantizar la fiabilidad y la validez de los resultados, especialmente en un contexto de investigación clínica, es imperativo seguir metodologías de validación rigurosas y prestar atención a la calidad de la señal (SQA, por sus siglas en inglés). La ausencia de un consenso claro sobre las bases de datos o métricas de evaluación para los detectores de QRS puede llevar a resultados que varían significativamente dependiendo de las condiciones del dataset, como los niveles de ruido y la colocación de los electrodos [1]. Por lo tanto, un investigador debe ser proactivo en la validación de las herramientas que utiliza.

La validación de un algoritmo de detección de QRS se basa típicamente en un proceso de comparación con un conjunto de anotaciones de referencia (ground truth), que generalmente provienen de marcas registradas cardiográficas. La metodología estándar implica pasar la señal a través del algoritmo de detección para obtener un conjunto de coordenadas de tiempo para los picos R detectados. Luego, estos picos detectados se comparan con los picos de referencia. La coincidencia se establece dentro de una ventana de tolerancia, comúnmente de 100 ms o 150 ms, lo que significa que un pico detectado se considera una detección correcta si se encuentra dentro de esa ventana de tiempo del pico de referencia [1] [14] [35]. Basándose en esta coincidencia, se calculan varias métricas clave. La sensibilidad (Se) o recall mide la proporción de latidos verdaderos que fueron correctamente detectados, mientras que el valor predictivo positivo (PP) o precision mide la proporción de picos detectados que eran latidos verdaderos (es decir, la tasa de falsos positivos) [14] [35]. El Error de Detección de Latido (DER, por sus siglas en inglés) se calcula como $(FP+FN)/(TP+FN)\times 100$, donde FP es el número de falsos positivos, FN es el número de falsos negativos y TP es el número de detecciones correctas [35]. Finalmente, el Error Cuadrático Medio a la Perfecta Detección (SDTP, por sus siglas en inglés) es una medida global que combina la sensibilidad y el valor predictivo [35].

Los benchmarks de rendimiento se realizan en bases de datos públicas y bien establecidas, como el MIT-BIH Arrhythmia Database, el MIT-BIH Normal Sinus Rhythm (NSR) Database, el European ST-T Database (EDB) y el MIT-BIH Noise Stress Test (NST) Database [11] [12] [28]. Cada una de estas bases de datos presenta desafíos únicos: algunas contienen registros de ritmos normales, otras arritmias específicas, y otras están diseñadas para probar la robustez de los algoritmos frente a diferentes tipos de ruido, como el de línea, el de movimiento o la deriva del punto de base [11] [12]. La evaluación en un conjunto diverso de bases de datos es crucial para determinar la generalización y robustez de un algoritmo. Por ejemplo, el detector HeartKey® demostró una alta sensibilidad y PP en bases de datos de referencia estándar, pero su rendimiento se mantuvo excepcionalmente alto incluso en bases de datos de electrodos secos (dry electrode), que simulan condiciones de grabación más desafiantes como las de los relojes inteligentes o los asientos de coche [14]. Esta diversidad de desafíos subraya la necesidad de evaluar las librerías en conjuntos de datos que sean pertinentes para el caso de uso específico del proyecto.

Además de la validación de la detección, el concepto de Calidad de la Señal (SQA) es fundamental. Una señal ECG de baja calidad puede llevar a falsos positivos o negativos, invalidando cualquier análisis posterior. Por lo tanto, es prudente implementar una fase de SQA antes de proceder con la detección y delineación. Existen varios enfoques para la SQA. Una aproximación simple consiste en verificar condiciones básicas como la presencia de un corazón latiendo (número de latidos dentro de un rango fisiológico, por ejemplo, 24-240 bpm) y la ausencia de artefactos graves como líneas planas (flatlines) o saturación [21] [27]. Una herramienta como `ECGAssess` implementa tres algoritmos de SQA en Python: un chequeo de señal estacionaria, un chequeo de frecuencia cardíaca usando el algoritmo de Pan-Tompkins para confirmar la presencia de latidos válidos, y un chequeo de relación señal-ruido (SNR) calculando la densidad espectral de potencia en la banda de señal (2-40 Hz) en comparación con las bandas de ruido (0-2 Hz y 40-250 Hz) [27]. Encontraron que un umbral de SNR de 0.5 dB funcionaba bien para distinguir entre señales aceptables e inaceptables [27].

Para una evaluación más sofisticada, se pueden calcular múltiples Índices de Calidad de la Señal (SQIs). La toolbox `ecgScorer` en MATLAB es un ejemplo excelente de esto, implementando un método que calcula hasta 37 SQIs agrupados en cuatro categorías: estadísticos, basados en la frecuencia, no lineales y basados en el detector de QRS [21]. Estos índices pueden incluir medidas como la kurtosis, la densidad espectral, la correlación entre latidos consecutivos, y la energía relativa

del complejo QRS [21]. El algoritmo final de `ecgScorer` toma estos índices como entrada para producir una puntuación final ('aScore') que indica la usabilidad clínica de la señal, logrando una alta sensibilidad (94.59%) y especificidad (98.38%) en la clasificación de señales buenas o malas [21]. Incluso librerías de Python como `neurokit2` y `PySiology` se mencionan como alternativas para la SQA, aunque `ecgScorer` destaca por su integración de un gran número de SQIs y un algoritmo de evaluación final [21]. La implementación de una fase de SQA rigurosa es una práctica de ingeniería de software crucial para cualquier sistema de análisis de ECG automatizado, ya que ayuda a identificar y descartar automáticamente datos de baja calidad, reduciendo el riesgo de conclusiones erróneas basadas en información deficiente.

Guía Práctica y Arquitectura de Pipeline para Investigadores y Desarrolladores Clínicos

Basado en el análisis exhaustivo de las herramientas y metodologías disponibles, se puede formular una guía práctica y una arquitectura de pipeline modular para que los investigadores y desarrolladores clínicos aborden eficazmente la tarea de análisis de ECG. La elección de las herramientas correctas no es una decisión única, sino un ejercicio de arquitectura de software que busca maximizar la precisión, la eficiencia y la robustez del flujo de trabajo. El enfoque recomendado es adoptar un enfoque híbrido, utilizando las fortalezas de Python para el desarrollo rápido y la gestión de datos, y complementándolo con las rutinas de cómputo de precisión y validación clínica de MATLAB cuando sea necesario.

Para el proceso fundamental de detección de R-picos, la elección de la librería de Python dependerá del contexto del proyecto. Para un flujo de trabajo de principio a fin que sea rápido de implementar y que ya incluya la delineación de ondas, la recomendación principal es comenzar con**`neurokit2`**. Su función `ecg_process()` simplifica enormemente el preprocesamiento, la detección y la delineación, y su capacidad para comparar métodos de delineación (como CWT y DWT) es invaluable para la investigación clínica [4]. Si, por el contrario, el proyecto prioriza la velocidad de procesamiento por encima de todo, como en sistemas de monitoreo en tiempo real o análisis de grandes cohortes de datos,**`sleepecg`**es la opción tecnológicamente superior debido a su implementación en C del algoritmo de Pan-Tompkins, que ofrece velocidades de procesamiento exponenciales [1]. Finalmente,

si el trabajo se centra en bases de datos estándar de PhysioNet y requiere la máxima adherencia a los estándares de la comunidad de investigación,**`wfdb-python`**es la elección más robusta y profesional, dada su integración completa con el ecosistema de WFDB y sus herramientas de post-procesamiento como `correct_peaks`[[3]].

Cuando el objetivo es la investigación comparativa y el control granular sobre el proceso algorítmico,**`py-ecg-detectors`**se erige como la herramienta ideal. Su capacidad para ejecutar y comparar directamente ocho algoritmos diferentes en el mismo conjunto de datos proporciona una perspectiva única y es una herramienta de investigación extremadamente poderosa [5] . Durante esta fase de investigación, es fundamental estudiar elalgoritmo Pan-Tompkinsy sus variantes [9] [15] . Entender su funcionamiento interno permite una validación más rigurosa de los resultados y una capacidad de diagnóstico superior cuando surgen problemas de rendimiento.

Para la delineación de ondas de alta precisión, que es crucial para mediciones clínicas como la duración del QRS, el enfoque debe cambiar hacia la inteligencia artificial. Dado que las librerías de Python tradicionales tienen limitaciones inherentes en este área, el camino hacia la máxima precisión pasa por el aprendizaje automático. Esto implicaría utilizar frameworks como PyTorch o TensorFlow para implementar modelos de deep learning personalizados [13] . Una estrategia más pragmática y rápida es explorar la posibilidad de integrar herramientas especializadas. El enfoque híbrido estratégico más prometedor es investigar la posibilidad de integrarBRAVEHEART, desarrollado en MATLAB, en el pipeline de Python utilizando elMATLAB Engine API. Esto permitiría aprovechar la delineación de alta precisión de BRAVEHEART, que utiliza redes neuronales LSTM, sin abandonar el ecosistema de Python para el resto del flujo de trabajo [18] [39] .

Antes de realizar cualquier análisis, es crucial implementar una fase de calificación de la calidad de la señal (SQA). Esta es una práctica de ingeniería de software fundamental para asegurar la fiabilidad de los resultados. Se puede empezar con una implementación básica de `ECGAssess` en Python para realizar comprobaciones de SNR y frecuencia cardíaca [27] . Para un análisis más exhaustivo, se puede utilizar la toolbox**`ecgScorer`**en MATLAB, accediéndola desde Python si es necesario, para obtener una puntuación final de usabilidad clínica basada en hasta 37 índices de calidad de la señal [21] .

En conclusión, la arquitectura de pipeline recomendada es modular y multifacética. Se recomienda construir un pipeline que pueda orquestar estas diferentes

herramientas. El flujo de trabajo podría comenzar en Python, utilizando una librería como `wfdb-python` para cargar datos de PhysioNet. Luego, la señal podría pasar por una fase de SQA utilizando una implementación de Python o MATLAB. Los R-picos podrían ser detectados con `neurokit2` para una delineación preliminar, y luego enviados a MATLAB para una validación final con `wqrs`. Finalmente, para obtener la delineación de ondas de máxima precisión, los R-picos se enviarían a BRAVEHEART a través del engine API, y los resultados finales se devolverían a Python para el análisis estadístico, la visualización y la generación de informes. Este enfoque híbrido maximiza la probabilidad de éxito para un investigador o desarrollador clínico, permitiéndoles navegar el complejo panorama de las herramientas de análisis de ECG con una estrategia informada y robusta.

Referencia

1. **ECG R peak detection in Python: a comparison of libraries** https://samproell.io/posts/signal/ecg-library-comparison/
2. **Aura-healthcare/Fast-QRS-detector** https://github.com/Aura-healthcare/Fast-QRS-detector
3. **Working with ECG Data in Python - In Digits** https://www.indigits.com/post/2022/10/ecg_python/
4. **Locate P, Q, S and T waves in ECG** https://neuropsychology.github.io/NeuroKit/examples/ecg_delineate/ecg_delineate.html
5. **py-ecg-detectors** https://pypi.org/project/py-ecg-detectors/
6. **Understanding ECG Signal Processing with Python** https://medium.com/@shahbaz.gondal588/understanding-ecg-signal-processing-with-python-b9dd4ea68682
7. **Waveform Database Software Package (WFDB) for Python** https://www.physionet.org/content/wfdb-python/3.3.0/wfdb/processing/qrs.py
8. **berndporr/py-ecg-detectors** https://github.com/berndporr/py-ecg-detectors
9. **Unsupervised ECG QRS Detection - AI4HEALTH** https://hooman650.github.io/ECG-QRS.html
10. **Pan-Tompkins Algorithm for Detecting QRS Waves** https://medium.com/@cosmicwanderer/pan-tompkins-algorithm-for-detecting-qrs-waves-29c5f2927906

11. **Performance Analysis of Ten Common QRS Detectors ...** https://pmc.ncbi.nlm.nih.gov/articles/PMC5964584/
12. **QRS Detection Based on Medical Knowledge and ...** https://www.mdpi.com/2076-3417/11/15/6995
13. **A simple and effective deep neural network based QRS ...** https://www.frontiersin.org/journals/physiology/articles/10.3389/fphys.2024.1384356/full
14. **A universal, high‐performance ECG signal processing ...** https://onlinelibrary.wiley.com/doi/10.1111/anec.12993
15. **A Real-Time QRS Detection Algorithm** https://www.robots.ox.ac.uk/~gari/teaching/cdt/A3/readings/ECG/Pan+Tompkins.pdf
16. **QRS detection using K-Nearest Neighbor algorithm (KNN) ...** https://www.sciencedirect.com/science/article/pii/S209012321200046X
17. **QRS Complex Detection and Measurement Algorithms for ...** https://pmc.ncbi.nlm.nih.gov/articles/PMC6231906/
18. **BRAVEHEART: Open-source software for automated ...** https://www.sciencedirect.com/science/article/pii/S0169260723004649
19. **BRAVEHEART: Open-source software for automated ...** https://www.medrxiv.org/content/10.1101/2023.05.17.23290060v1.full-text
20. **Is there any ready made tool box available to analyze ECG ...** https://www.researchgate.net/post/Is-there-any-ready-made-tool-box-available-to-analyze-ECG-signal
21. **An open source MATLAB toolbox for ECG signal quality ...** https://www.sciencedirect.com/science/article/pii/S235271102400270X
22. **Open Source ECG Analysis** https://www.cinc.org/archives/2002/pdf/101.pdf
23. **translating matlab ecg signal quality algorithms into python: ...** https://repository.gatech.edu/bitstreams/939f40ac-3733-446f-ac51-08b066fdc03f/download
24. **The Best MATLAB Toolboxes for Biosignal Processing** https://www.linkedin.com/pulse/best-matlab-toolboxes-biosignal-processing-pedro-gomes-dm3ke
25. **Python and Biosignals: Use Cases and Best Practices** https://www.proxet.com/blog/biosignals-processing-in-python-best-ways-for-its-implementation
26. **ecg-signal · GitHub Topics** https://github.com/topics/ecg-signal?l=matlab/1000
27. **A Python-Based Toolbox to Assess ECG Lead Signal Quality** https://www.researchgate.net/publication/360420951_ECGAssess_A_Python-Based_Toolbox_to_Assess_ECG_Lead_Signal_Quality

28. **Automatic Real-Time Embedded QRS Complex Detection for ...** https://pmc.ncbi.nlm.nih.gov/articles/PMC4848097/
29. **A real-time QRS detection algorithm for the Arduino platform** https://github.com/blakeMilner/real_time_QRS_detection
30. **Deploying the ECG Pan-Tompkins filter cascade on an STM32 ...** https://www.advsolned.com/wp-content/uploads/2021/10/ASN-AN029.pdf
31. **Real-Time ECG QRS Detection on ARM Cortex-M Processor** https://www.mathworks.com/help/ecoder/armcortexm/ref/Real-Time-ECG-QRS-Detection-on-ARM-Cortex-M-Processor-example.html
32. **A real-time embedded system to detect QRS-complex and ...** https://dl.acm.org/doi/10.1016/j.eswa.2022.119221
33. **An Embedded System for Real-Time Atrial Fibrillation ...** https://www.mdpi.com/2673-4117/5/4/143
34. **Real-Time ECG QRS Detection - MATLAB & Simulink** https://www.mathworks.com/help/dsp/ug/real-time-ecg-qrs-detection.html
35. **Demonstration of QRS detector - File Exchange - MATLAB ...** https://www.mathworks.com/matlabcentral/fileexchange/164521-demonstration-of-qrs-detector?s_tid=blogs_rc_5
36. **ecg-kit - File Exchange - MATLAB Central** https://www.mathworks.com/matlabcentral/fileexchange/50769-ecg-kit?WxqRl=LlI2ymhT3V
37. **Integrate MATLAB with External Programming Languages ...** https://www.mathworks.com/help/matlab/matlab_external/integrate-matlab-with-external-programming-languages-and-systems.html
38. **Loading Compiled Matlab Shared Library in Python Using ...** https://stackoverflow.com/questions/31375758/loading-compiled-matlab-shared-library-in-python-using-ctypes
39. **Using MATLAB with Python** https://nl.mathworks.com/products/matlab/matlab-and-python.html
40. **Using MATLAB with Python** https://www.mathworks.com/videos/using-matlab-with-python-1591216182793.html
41. **Using MATLAB and Python Resources** https://www.mathworks.com/products/matlab/getting-started/using-matlab-python.html