# Continuum: 단절된 자아 연결을 위한 AI 기반 애플리케이션의 기술 아키텍처 및 구현 로드맵

## 섹션 1: 아키텍처 철학 - 앰비언트 프라이빗 컴패니언

제안된 'Continuum' 애플리케이션의 구축을 위한 모든 기술적 결정은 명확하고 타협 불가능한 철학적 기반 위에서 이루어져야 합니다. 사용자의 비전을 기술적 원칙으로 변환하는 이 섹션에서는, 우리가 구축하고자 하는 시스템의 핵심적인 '왜'와 '어떻게'를 정의합니다. 특히, 이 애플리케이션이 개인적 용도로 사용된다는 제약 조건은, 역설적으로 최신 하드웨어와 소프트웨어의 발전에 힘입어 점점 더 실현 가능해지고 있는 강력한 온디바이스(on-device) 패러다임을 채택할 수 있는 절호의 기회를 제공합니다.

### 1.1 핵심 원칙 정의

모든 후속 아키텍처 결정의 기준이 될 네 가지 핵심 원칙은 다음과 같습니다.

* **마찰 없는 포착 (Zero-Friction Capture):** 시스템은 데이터 입력의 인지적 부하를 최소화하는 것을 최우선으로 해야 합니다. 모든 입력 메커니즘은 메시지를 보내거나 생각을 소리 내어 말하는 것만큼이나 원활해야 합니다. 사용자가 정보를 기록하기 위해 의식적인 노력을 기울여야 한다면, 시스템의 핵심 가치는 훼손됩니다.
* **선제적 종합 (Proactive Synthesis):** 이 애플리케이션의 주요 가치는 수동적인 데이터베이스가 아니라 능동적인 종합자(synthesizer)로서의 역할에 있습니다. 시스템은 명시적인 질의 없이도 사용자의 요구를 예측하고 연결점을 표면화해야 합니다. 이는 상황 인식(context-aware) 기반의 백그라운드 처리에 대한 상당한 투자를 필요로 합니다.
* **설계 기반 프라이버시 (Privacy by Design - 온디바이스 최우선):** 원시 텍스트부터 파생된 벡터 임베딩에 이르기까지 모든 사용자 데이터는 예외 없이 사용자의 기기 내에서만 저장되고 처리될 것입니다. 사용자 생성 콘텐츠는 어떠한 처리 목적으로도 제3자 서버로 전송되지 않습니다. 이는 제한 사항이 아니라 핵심 기능입니다.1 이 접근 방식은 본질적으로 강력한 보안과 프라이버시를 제공하며, 사용자가 자신의 가장 내밀한 생각을 신뢰하고 기록할 수 있는 기반이 됩니다.
* **기능으로서의 성능 (Performance as a Feature):** 온디바이스 AI는 계산 비용이 많이 듭니다.3 아키텍처는 모바일 환경의 제약(배터리, 메모리, CPU/NPU)에 맞춰 무자비하게 최적화되어야 합니다. 애플리케이션이 반응성이 뛰어나고 사용자의 기기 경험을 저하시키지 않도록 보장하는 것은 필수적입니다.

### 1.2 핵심 과제: 성능과 효율성의 균형

이 프로젝트의 중심적인 엔지니어링 문제는 정교한 AI 파이프라인(임베딩, 인덱싱, 검색 증강 생성(RAG))을 모바일 기기라는 자원 제약적인 환경 내에서 구현하는 것입니다. 이 과제는 이후 섹션에서 논의될 기술 스택의 모든 구성 요소에 대한 상세한 트레이드오프 분석의 배경이 됩니다.

이 프로젝트의 '비상업적' 요구 사항은 가장 큰 기술적 이점입니다. 이는 아키텍처를 클라우드 인프라, 멀티테넌시, 서버 측 AI 비용의 복잡성으로부터 해방시켜, 온디바이스 경험 최적화에만 집중할 수 있게 합니다. 상업용 앱은 종종 성능과 확장성을 위해 OpenAI API와 같은 클라우드 AI에 의존하며, 이는 지연 시간, 비용, 그리고 심각한 프라이버시 문제를 야기합니다. 그러나 'Continuum'은 상업적 비즈니스 모델의 필요성을 제거함으로써 클라우드 기반 AI 처리의 필요성 자체를 없앱니다. 최신 연구는 성능 좋은 온디바이스 AI 프레임워크와 모델로의 강력하고 가속화되는 추세를 보여줍니다.1 따라서 우리는 대부분의 상업용 대안보다 더 빠르고(네트워크 지연 없음), 저렴하며(API 비용 없음), 더 사적인(Privacy by Design) 시스템을 설계할 수 있습니다. 이는 개인용 앱임에도 불구하고, 사용자의 내밀한 '두 번째 뇌'라는 비전과 완벽하게 일치하는 강력한 특징이 됩니다.

## 섹션 2: 핵심 상호작용 루프의 해부: 포착, 연결, 구체화

이 섹션에서는 앱의 근본적인 사용자 여정을 세분화하여 각 단계에 대한 기술적 요구사항과 구현 경로를 정의합니다. '포착(Capture)', '연결(Connect)', '구체화(Crystallize)'라는 세 단계는 'Continuum'의 핵심 가치를 실현하는 순환적인 프로세스를 형성합니다.

### 2.1 포착 엔진: 다중 모드, 유비쿼터스 입력 시스템

사용자의 삶의 조각들을 원활하게 수집하기 위해, 포착 엔진은 다양한 형태의 입력을 최소한의 노력으로 처리해야 합니다.

* **음성 "생각 스트리밍":** 사용자가 길게 누르고 말하는 동안 실시간으로 음성을 텍스트로 변환하는 기능을 구현합니다. 이는 네이티브 OS의 온디바이스 음성 인식 기능을 활용하여 이루어집니다. 예를 들어, 안드로이드의 Live Transcribe 기능이나 관련 API 7, 또는 구글의 Speech-to-Text API가 제공하는 온프레미스 옵션 8, 그리고 애플의 온디바이스 받아쓰기 프레임워크가 이 기능의 기반이 될 수 있습니다. 목표는 '생각을 기록하기 위해 길게 누르는' 직관적인 인터페이스를 제공하는 것입니다.
* **시스템 전반의 통합 (공유):** 마찰을 줄이는 데 있어 가장 중요한 기능 중 하나입니다.
  + **iOS:** iOS 공유 확장(Share Extension)을 구현합니다. 하지만 연구에 따르면 약 120MB의 엄격한 메모리 제한이 존재하여, 긴 음성 메모나 고해상도 이미지와 같은 대용량 데이터를 처리할 때 문제가 될 수 있습니다.9 우리의 아키텍처는 이 문제를 해결하기 위해 두 가지 전략 중 하나를 채택해야 합니다. 첫째, 확장 내에서 데이터를 청크 단위로 처리하거나, 둘째, 공유된 항목에 대한 참조(예: 파일 경로)를 공유 앱 그룹 컨테이너에 저장하고, 메인 앱이 백그라운드에서 임베딩 및 인덱싱과 같은 무거운 작업을 수행하도록 트리거하는 것입니다. 구현 세부 사항은 관련 문서에서 확인할 수 있습니다.11
  + **Android:** ACTION\_SEND 인텐트 필터를 구현합니다. 이는 다른 앱으로부터 공유된 데이터를 수신하는 표준적이고 강력한 메커니즘입니다.12 이 시스템은 백그라운드 처리와 관련하여 일반적으로 iOS보다 더 유연한 환경을 제공합니다.
* **자동화된 데이터 수집:** 진정으로 연속적인 서사를 구축하기 위해, 앱은 주요 이벤트를 수동적으로 기록해야 합니다. 이는 가능한 경우 API 통합을 통해 이루어지며, 섹션 5에서 자세히 설명할 컨텍스트 API를 활용하여 "사무실 도착" 또는 "5km 달리기 완료"와 같은 로그 항목을 자동으로 생성함으로써 달성될 수 있습니다.

### 2.2 연결 패브릭: 온디바이스 벡터 데이터베이스

이것은 애플리케이션의 심장부로, 포착된 모든 생각과 경험의 벡터 임베딩을 저장합니다. 데이터베이스의 선택은 성능과 확장성에 매우 중요하며, 가볍고 빠르며 선택된 모바일 플랫폼과의 네이티브 통합을 제공해야 합니다.

* **후보 분석:** 주요 온디바이스 벡터 데이터베이스 솔루션을 평가합니다.
  + **ObjectBox:** "안드로이드/자바/코틀린 및 iOS/스위프트를 위한 최초의 온디바이스 벡터 데이터베이스"로 명시적으로 마케팅되는 강력한 후보입니다.15 빠른 ANN 검색을 위해 HNSW 알고리즘을 사용하며, 개발을 단순화하는 성숙한 객체 지향 API를 제공합니다.17 코틀린 20 및 스위프트 17에 대한 튜토리얼이 제공됩니다.
  + **Qdrant Edge:** 임베디드 및 온디바이스 사용 사례를 위해 특별히 설계된 새롭고 유망한 솔루션입니다.25 인프로세스 라이브러리로 실행되므로 애플리케이션이 인덱싱 및 검색 작업을 완벽하게 제어할 수 있어 모바일에서 리소스 소비를 관리하는 데 이상적입니다. 현재 비공개 베타 버전이지만 최첨단 기술을 대표합니다.
  + **Milvus Lite:** 인기 있는 Milvus 벡터 데이터베이스의 경량 버전으로, 노트북 및 엣지 디바이스에서 실행되도록 설계되었습니다.26  
    pymilvus 파이썬 SDK에 포함되어 있어 직접적인 네이티브 통합이 복잡할 수 있지만 가능성은 있습니다. 분산 버전에 비해 인덱스 지원이 제한적입니다(FLAT, IVF\_FLAT).
  + **ChromaDB:** 서버 환경에서 자주 사용되지만, Chroma는 영구 저장소와 함께 로컬에서 실행할 수 있습니다.28 주요 클라이언트는 파이썬과 자바스크립트이므로 리액트 네이티브 구현에 유리합니다. 개발자 친화적이고 시작하기 쉽습니다.30
  + **SQLite 확장 (예: MobileRAG):** MobileRAG 논문은 메타데이터와 매핑을 저장하기 위해 표준 SQLite 데이터베이스를 사용하고, 메모리 효율적인 맞춤형 벡터 검색 알고리즘(EcoVector)이 인덱싱 및 검색을 처리하는 새로운 접근 방식을 제안합니다.31 이는 더 많은 맞춤화가 필요하지만 잠재적으로 고도로 최적화된 솔루션을 나타냅니다.

벡터 데이터베이스는 앱의 중앙 '연결 패브릭'입니다. 그 성능은 사용자의 검색 및 발견 경험에 직접적인 영향을 미칩니다. 모바일 기기에서 성능은 속도뿐만 아니라 리소스 효율성(RAM, 저장 공간, 배터리)에 관한 것이기도 합니다. 빠르지만 배터리를 소모시키는 데이터베이스는 실패작입니다. 각 후보는 서로 다른 강점을 가지고 있습니다. ObjectBox는 성숙하고 모바일 네이티브이며 16, Qdrant Edge는 엣지 환경을 위해 특별히 제작되었습니다.25 Chroma는 개발자 친화적이지만 모바일 네이티브는 덜하며 28, Milvus Lite는 서버 솔루션의 축소 버전입니다.26 맞춤형 솔루션은 최대의 제어력을 제공합니다.31 아래의 의사 결정 매트릭스는 성능, 리소스 사용, 네이티브 통합 용이성이라는 모바일 특화된 핵심 축에 걸쳐 이러한 옵션들을 객관적으로 비교하는 유일한 방법입니다. 이 표는 앱 아키텍처에서 가장 중요한 구성 요소 선택에 대한 데이터 기반의 토대를 제공할 것입니다.

**표 1: 온디바이스 벡터 데이터베이스 결정 매트릭스**

| 기준 | ObjectBox | Qdrant Edge | Milvus Lite | Chroma (로컬) | 맞춤형 (SQLite+EcoVector) |
| --- | --- | --- | --- | --- | --- |
| **주요 언어/SDK** | Swift, Kotlin/Java | Rust (라이브러리) | Python | Python, JS | C++, Java (네이티브) |
| **인덱싱 알고리즘** | HNSW | HNSW | FLAT, IVF\_FLAT | HNSW | 맞춤형 (EcoVector) |
| **CRUD 성능** | 매우 높음 15 | 벤치마크 필요 | 벤치마크 필요 | 중간 | 벤치마크 필요 |
| **쿼리 지연 시간** | 매우 낮음 15 | 매우 낮음 (예상) 25 | 낮음 (제한적 인덱스) 26 | 낮음 | 매우 낮음 (최적화 시) 31 |
| **메모리 사용량** | 낮음 | 매우 낮음 (설계 목표) 25 | 중간 | 중간 | 매우 낮음 (설계 목표) 31 |
| **바이너리 크기** | 작음 | 매우 작음 (예상) | 중간 | 중간 | 작음 |
| **라이선스** | Apache 2.0 / 상업용 | Apache 2.0 | Apache 2.0 | Apache 2.0 | 연구용 / 맞춤형 |
| **온디바이스 성숙도** | 높음 | 낮음 (베타) | 낮음 | 낮음 | 연구 단계 |

### 2.3 구체화 표면: 방해 없이 통찰력 제시하기

이 구성 요소는 발견의 사용자 경험에 관한 것입니다. UI는 AI가 생성한 연결("인사이트 스레드", "맥락적 회상")을 도움이 되면서도 방해되지 않는 방식으로 표면화해야 합니다.

* **고급 UI 패턴:** 단순한 알림을 넘어선 패턴을 채택합니다.
  + **동적 홈 화면 위젯:** 이는 선제적 통찰력을 위한 최고의 공간입니다. iOS 18 33 및 안드로이드 15(위젯 미리보기를 개인화하고 Glance API를 사용함 35)의 최신 위젯 기능을 활용할 것입니다. 위젯은 "작년 오늘 이 생각"이나 발전 중인 "인사이트 스레드"의 요약과 같은 단일하고 관련성 높은 정보를 표시할 수 있습니다. 배터리 보존을 위해 OS가 부과하는 위젯 업데이트 빈도 제한을 염두에 두어야 합니다.36
  + **비간섭적 인앱 신호:** 방해가 되는 팝업 대신, "스포트라이트 효과"나 "맥동 애니메이션"과 같은 미묘한 패턴을 사용하여 사용자가 앱과 상호작용할 때 새로 형성된 연결에 주의를 끌 것입니다.39 예를 들어, 사용자가 새 노트를 입력할 때 하단에 작고 해제 가능한 배너가 나타나 "이것은 '프로젝트 X'에 대한 당신의 생각과 관련이 있는 것 같습니다"라고 제안할 수 있습니다.40
  + **예측적 뒤로 가기 제스처 통합 (Android):** 안드로이드 15 이상에서는 앱이 새로운 예측적 뒤로 가기 제스처를 지원하여 42 더 유동적이고 통합된 OS 경험을 제공할 것입니다.

## 섹션 3: 온디바이스 AI 스택: Continuum 엔진 심층 분석

이 섹션은 보고서의 기술적 핵심으로, AI 파이프라인의 모든 구성 요소 선택과 통합을 상세히 설명하여 전체가 기기 내에서 효율적으로 실행될 수 있도록 보장합니다.

### 3.1 기반: 올바른 개발 패러다임 선택 (네이티브 vs. 크로스플랫폼)

이는 가장 중요한 아키텍처 결정으로, 성능, OS 기능 접근성, 개발 속도에 영향을 미칩니다.

* **네이티브 (Swift/Kotlin):** 최고의 성능과 하드웨어 가속기(NPU, GPU) 및 최신 OS API에 대한 직접적인 접근을 제공합니다.44 이는 계산 집약적인 AI 작업과 상황 인식 및 고급 위젯에 필요한 깊은 OS 통합에 매우 중요합니다. 스위프트와 코틀린은 성숙하고 안전하며 각 플랫폼에서 선호되는 언어입니다.
* **크로스플랫폼 (React Native):** 새로운 아키텍처(JSI, TurboModules, Fabric)는 네이티브와의 성능 격차를 크게 줄였습니다.46 단일 코드베이스를 허용하여 개인 개발자나 소규모 팀에게 매력적입니다. 그러나 최신 OS 기능 지원에 지연이 있을 수 있으며, 온디바이스 AI를 위한 전문 네이티브 라이브러리(예: Core ML, MediaPipe)에 접근하려면 브리지를 구축해야 하므로 복잡성이 추가됩니다.47 MediaPipe 48 및 Core ML 52에 대한 래퍼가 존재하지만, 직접적인 네이티브 통합만큼 최적화되지 않을 수 있습니다.

Continuum의 개발 플랫폼 선택은 다른 모든 스택 구성 요소에 영향을 미치는 가장 중요한 결정입니다. Continuum은 표준적인 CRUD 앱이 아닙니다. 핵심 기능(온디바이스 AI, 선제적 상황 감지)은 기본 운영 체제의 기능과 깊이 연결되어 있습니다. 리액트 네이티브의 발전은 상당하여 46 실행 가능한 옵션이 되었습니다. 그러나 네이티브 플랫폼은 항상 새로운 하드웨어(NPU) 및 소프트웨어(최신 컨텍스트 API, 위젯 프레임워크)에 대한 자사 최적화된 접근성을 가질 것입니다.44 아래 표는 Continuum의 특정 요구 사항에 기반한 구조화된 비교를 강제합니다. 예를 들어, '백그라운드 처리 신뢰성' 항목에서는 네이티브

CLBackgroundActivitySession 55의 견고성과 자바스크립트 브리지를 통해 백그라운드 작업을 관리하는 잠재적 복잡성을 분석합니다. 이 구조화된 분석은 제품 비전의 장기적인 성공과 고유한 요구 사항을 우선시하는 명확하고 방어 가능한 권장 사항으로 이어질 것입니다.

**표 2: 플랫폼 아키텍처 트레이드오프 (네이티브 vs. 리액트 네이티브)**

| 기준 | 네이티브 (Swift/Kotlin) | 리액트 네이티브 (New Architecture) | Continuum을 위한 정당성 및 권장 사항 |
| --- | --- | --- | --- |
| **AI/ML 성능 (GPU/NPU 접근)** | 직접적, 최적화됨 | JSI를 통해 가능하나 간접적 | **네이티브 권장.** 온디바이스 AI는 성능이 가장 중요하며, 하드웨어 가속기에 대한 직접 접근은 지연 시간을 줄이고 배터리 수명을 보존하는 데 필수적입니다. |
| **백그라운드 처리 신뢰성** | 강력하고 OS에 의해 관리됨 | 복잡하며 OS 제한에 더 취약함 | **네이티브 권장.** 앱의 '앰비언트' 특성은 신뢰할 수 있는 백그라운드 컨텍스트 감지에 달려 있으며, 이는 네이티브 API를 통해 가장 잘 달성됩니다. |
| **컨텍스트 인식 API 접근** | 최신 API에 즉시 접근 가능 | 래퍼 또는 브리지가 필요하며 지연될 수 있음 | **네이티브 권장.** 선제적 제안은 최신 OS 수준의 컨텍스트 API(예: Awareness API, Core Motion)에 대한 즉각적인 접근을 요구합니다. |
| **고급 UI (위젯/실시간 현황)** | 완전한 기능, 최고의 성능 | 제한적일 수 있으며 네이티브 모듈이 필요함 | **네이티브 권장.** 동적 위젯은 '구체화' 단계의 핵심 표면이며, 네이티브 구현은 가장 풍부하고 반응성이 뛰어난 경험을 제공합니다. |
| **개발 속도** | 플랫폼별로 느림 | 단일 코드베이스로 빠름 | **트레이드오프.** 리액트 네이티브가 초기 MVP에는 더 빠를 수 있지만, 핵심 기능에 필요한 네이티브 브리지의 복잡성이 이 이점을 상쇄할 수 있습니다. |
| **장기 유지보수성** | 플랫폼별로 명확함 | 프레임워크 및 종속성 업데이트에 민감함 | **네이티브 권장.** OS와 긴밀하게 통합된 앱의 경우, 네이티브 스택은 장기적으로 더 적은 추상화 계층과 더 예측 가능한 유지보수 경로를 제공합니다. |

### 3.2 임베딩 엔진: 삶을 벡터로 변환하기

임베딩의 품질은 시맨틱 검색의 품질을 직접적으로 결정합니다. 모델은 모바일에서 효율적으로 실행될 만큼 작으면서도 미묘한 의미를 포착할 만큼 강력해야 합니다.

* **후보 분석:**
  + **MediaPipe Text Embedder:** Universal Sentence Encoder와 같은 모델을 활용하는 즉시 사용 가능한 작업을 제공합니다.56 구글의 AI Edge 제품군의 일부로 크로스플랫폼을 지원하며 잘 지원되는 옵션입니다.1
  + **Sentence-Transformers (ONNX):** all-MiniLM-L6-v2와 같은 모델은 검색 작업에서 높은 성능으로 평가받고 있습니다.57 ONNX 형식으로 변환하여 ONNX Runtime을 사용해 모바일에서 실행할 수 있으며 58, 이는 iOS(Core ML)와 안드로이드(NNAPI) 모두에서 하드웨어 가속을 지원합니다.60 이는 고품질 오픈소스 임베딩을 위한 경로를 제공합니다.
  + **네이티브 임베더 (Apple의 NLEmbedding):** 애플은 단어 임베딩 생성을 위한 네이티브 프레임워크를 제공합니다.61 편리하지만, 이는 종종 단어 수준이며 시맨틱 검색 작업에는 최신 문장 수준 트랜스포머 모델만큼 강력하지 않을 수 있습니다.62
* **성능 고려사항:** 시맨틱 품질(예: MTEB 점수)뿐만 아니라 63, 대상 모바일 하드웨어에서의 추론 지연 시간과 메모리 사용량도 벤치마킹해야 합니다.3

### 3.3 종합 코어: 온디바이스 RAG 및 생성 모델

여기서는 검색 증강 생성(RAG) 파이프라인을 로컬에서 구현하는 방법을 상세히 설명합니다. 이는 검색된 데이터를 일관된 통찰력으로 종합하는 '구체화' 단계입니다.

* **경량 LLM 선택:** 수용 가능한 성능으로 모바일 기기에서 실행할 수 있는 작고 효율적인 LLM이 필요합니다.
  + **후보:** 구글의 Gemma 2B 65, 마이크로소프트의 Phi-3/Phi-2 66, 그리고 10억-30억 파라미터 범위의 다른 모델들이 있습니다.68 벤치마크에 따르면 Phi-2가 약간 더 큼에도 불구하고 일부 작업에서는 Gemma 2B보다 성능이 뛰어날 수 있습니다.66
* **온디바이스 추론 프레임워크:** LLM을 실행하는 엔진입니다.
  + **MediaPipe LLM Inference API:** 안드로이드에서 Gemma 및 Phi-2를 포함한 일부 LLM을 온디바이스로 실행하기 위한 구글의 고수준 API입니다.65 고급 기기에 최적화되어 있으며 LoRA 사용자 정의를 지원합니다.
  + **MLC LLM:** Llama 3를 포함한 다양한 모델을 맞춤형 C++ 런타임을 통해 iOS 및 안드로이드를 포함한 여러 플랫폼에 배포할 수 있는 다목적 머신러닝 컴파일러입니다.5 심층적인 최적화 기능을 제공합니다.
  + **기타 프레임워크:** TensorFlow Lite(현 LiteRT) 1 및 PyTorch Mobile 71은 기본적인 런타임이지만 복잡한 LLM을 통합하려면 더 많은 수작업이 필요할 수 있습니다.
* **RAG 파이프라인 구현:** 온디바이스 실행에 맞게 조정된 표준 RAG 패턴을 따릅니다.72 완전한 온디바이스 RAG 파이프라인은 학술적으로 시연되었으며(MobileRAG31), 안드로이드용 구글 AI Edge RAG SDK와 같은 SDK에서 지원됩니다.74 프로세스는 다음과 같습니다:
  1. 사용자 쿼리는 3.2의 엔진을 사용하여 임베딩됩니다.
  2. 2.2의 벡터 DB를 쿼리하여 상위 k개의 가장 유사한 과거 항목을 찾습니다.
  3. 이 항목들은 원본 쿼리와 함께 컨텍스트로 3.3의 온디바이스 LLM에 대한 프롬프트에 전달됩니다.
  4. LLM은 사용자 자신의 데이터에 기반한 종합적인 응답을 생성합니다. LlamaIndex 75 및 LangChain 80과 같은 도구를 사용한 이 흐름에 대한 튜토리얼이 존재하며, 프레임워크 자체가 직접적인 모바일 배포에는 너무 무거울 수 있지만 귀중한 패턴을 제공합니다.

## 섹션 4: 고급 검색 및 종합 전략

이 섹션에서는 광범위한 개념적 회상과 구체적이고 문자 그대로의 검색 모두에 대한 요구를 직접적으로 해결하며, AI의 통찰력 품질을 단순한 시맨틱 검색 이상으로 끌어올리는 기술을 소개합니다.

### 4.1 시맨틱 검색을 넘어서: 하이브리드 검색 구현

* **문제점:** 시맨틱(벡터) 검색은 개념적으로 유사한 아이디어("시장 변동성에 대한 느낌")를 찾는 데는 뛰어나지만, 특정 키워드나 개체("TSLA 주식 아이디어")를 검색하는 데는 실패할 수 있습니다.83
* **해결책:** 벡터 검색 결과를 BM25와 같은 전통적인 키워드 기반 검색 알고리즘과 결합하는 하이브리드 검색을 구현할 것입니다.84 BM25는 무거운 라이브러리 없이 구현할 수 있는 가볍고 효과적인 알고리즘입니다.86 이 이중 접근 방식은 시스템이 "분위기"와 "사실"을 모두 검색할 수 있도록 보장합니다.

### 4.2 지능형 순위 지정: 상호 순위 융합(RRF)을 통한 결과 융합

* **과제:** 벡터 검색과 BM25에서 나온 두 개의 순위 목록을 지능적으로 병합할 방법이 필요합니다. 점수가 서로 다른 척도에 있기 때문에 단순한 점수 정규화는 종종 비효율적입니다.88
* **알고리즘:** 상호 순위 융합(Reciprocal Rank Fusion, RRF)은 순위 목록을 병합하기 위한 간단하고 강력하며 튜닝이 필요 없는 알고리즘입니다.90 각 목록에서 문서의 순위 역수를 기반으로 새 점수(  
  score=1/(k+rank))를 계산합니다. 이 방법은 *여러* 결과 집합에서 상위에 나타나는 문서를 우선시하여 더 강력한 최종 순위를 만듭니다.93 하이브리드 검색 결과를 LLM에 전달하기 전에 온디바이스에서 RRF를 구현하여 순위를 재조정할 것입니다.

### 4.3 프롬프트의 기술: 개인적 종합을 위한 엔지니어링

* **목표:** 최종 단계는 온디바이스 LLM이 검색된 데이터로부터 고품질의 미묘한 통찰력을 생성하도록 유도하는 것입니다. 이를 위해서는 정교한 프롬프트 엔지니어링이 필요합니다.
* **요약 기법:** 여러 검색된 노트를 요약하기 위해 "맵리듀스(Map-Reduce)" 전략을 사용할 것입니다. 검색된 각 노트("문서")는 개별적으로 요약된 다음, 최종 프롬프트가 LLM에게 이러한 개별 요약을 일관된 서사로 종합하도록 요청합니다.96 원하는 형식, 길이 및 초점을 지정하는 구조화된 프롬프트를 사용할 것입니다.98
* **ReAct 프레임워크:** 다단계 추론이 필요한 더 복잡한 쿼리(예: "재생 에너지 투자에 대한 내 생각이 어떻게 발전했으며, 나에게 영향을 준 주요 기사는 무엇이었는가?")를 위해 ReAct(Reason-Act) 프롬프팅 프레임워크를 구현할 것입니다.102 LLM은 "생각"(예: "재생 에너지에 대한 모든 노트를 찾아야겠다"), "행동"(예:  
  search("재생 에너지 투자")), 그리고 "관찰"(해당 검색의 결과)을 생성하도록 프롬프트됩니다. 이 루프는 모델이 복잡한 문제를 분해하고 사용자의 지식 기반을 도구로 사용하여 상호작용하도록 하여 더 정확하고 감사 가능한 통찰력을 이끌어냅니다.103

'구체화' 단계의 품질은 LLM의 성능에만 의존하는 것이 아니라, 검색된 컨텍스트의 품질에 결정적으로 영향을 받습니다. "멍청한" 검색 시스템은 LLM에 쓰레기를 제공하고, 결과적으로 쓰레기 같은 통찰력을 낳습니다. 따라서 고급 검색 파이프라인(하이브리드 검색 + RRF)에 투자하는 것은 최종 사용자 경험을 극적으로 향상시키는 높은 레버리지 활동입니다. 사용자의 핵심 문제는 단편화된 자아를 연결하는 것이며, 이는 서로 다른 생각을 연결해야 함을 의미합니다. 일부 생각은 개념적("미래에 대한 불안감")이고 다른 생각은 구체적("NVDA 주식 매수")입니다. 순수 벡터 검색은 전자에 뛰어나지만 후자에는 실패할 수 있습니다.83 순수 키워드 검색은 그 반대입니다. 이는 사용자의 마음을 완벽하게 파악하기 위해 검색 시스템이 두 가지 모드 모두에 능숙해야 함을 시사합니다. 하이브리드 검색은 이 이중 기능을 제공합니다.85 그러나 하이브리드 검색은 병합해야 하는 두 개의 개별 결과 목록을 생성하며, 순진한 병합은 문제가 있습니다.88 RRF는 이 목록들을 융합하기 위한 수학적으로 건전하고 가벼운 방법을 제공하여,

*두* 검색 패러다임에서 가장 관련성 높은 항목이 우선 순위를 갖도록 보장합니다.91 따라서 하이브리드 검색 + RRF 파이프라인을 구현하는 것은 사용자의 핵심 심리적 문제에 대한 직접적인 아키텍처적 대응이며, LLM이 완전한 서사를 종합하기 위해 가장 전체적으로 관련성 있는 컨텍스트를 받도록 보장합니다.

## 섹션 5: 선제적 시스템: 사용자의 세계를 감지하고 반응하기

이 섹션에서는 Continuum의 '마법', 즉 사용자의 상황을 이해하여 적시에 적절한 정보를 표면화하는 능력의 구현을 상세히 설명합니다.

### 5.1 감각 계층: 기기 및 환경 컨텍스트 활용

이 계층은 배터리 효율이 매우 높은 방식으로 상황 신호를 수집하는 역할을 합니다. GPS와 같은 고전력 센서를 지속적으로 폴링하는 것을 피해야 합니다.

* **iOS:** 전력 소모를 줄이기 위해 지속적인 추적이 아닌, 중요 위치 변경 모니터링 및 지역 모니터링(지오펜싱)을 위해 Core Location 프레임워크를 사용할 것입니다.55 활동 감지(걷기, 달리기, 자동차)를 위해서는 저전력, 장기 모니터링을 위해 설계된  
  CMMotionActivityManager를 사용할 것입니다.110 또한 앱이 일시 중단되었을 때 이러한 이벤트를 처리하기 위해 백그라운드 작업 실행을 올바르게 구성해야 합니다.55
* **Android:** **Google Awareness API**가 이상적인 도구입니다. 이 API는 위치, 활동, 헤드폰 등 여러 컨텍스트 신호를 단일의 전력 효율적인 API로 통합합니다.113 우리는  
  Fence API를 사용하여 앱이 백그라운드에 있을 때에도 콜백을 트리거할 컨텍스트 조건의 조합(예: 사용자가 걷고 있음 AND 헤드폰이 연결됨)을 등록할 것입니다. 이는 지속적인 추적을 위해 FusedLocationProvider를 직접 사용하는 것보다 훨씬 효율적입니다.115

### 5.2 트리거 로직: 신호에서 통찰력으로

이것은 선제적 시스템의 두뇌입니다. 상황 신호(예: "사용자가 [사무실 위치]에 도착함") 자체는 유용하지 않습니다. 지능적인 쿼리로 변환되어야 합니다.

* **아키텍처:** 간단한 규칙 엔진을 설계할 것입니다. 컨텍스트 펜스가 트리거되면(예: Awareness API 콜백), 시스템은 다음을 수행합니다:
  1. 컨텍스트 식별: event: ARRIVED, location\_tag: OFFICE.
  2. 규칙 엔진 참조: "event가 OFFICE에서 ARRIVED일 때, WORK 태그가 있거나 과거에 이 위치에서 생성된 노트에 대한 쿼리를 공식화하라."
  3. 2.2의 벡터 데이터베이스에 대해 쿼리 실행.
  4. 관련 결과가 발견되면, "사용자가 방금 사무실에 도착했습니다. 이 관련 과거 생각과 프로젝트를 요약해 주세요."와 같은 프롬프트와 함께 3.3의 온디바이스 LLM에 전달.
  5. 홈 화면 위젯이나 미묘한 알림과 같은 비간섭적인 UI 요소(2.3)를 통해 사용자에게 종합된 통찰력을 제시합니다. 이는 iOS 117 및 안드로이드의 선제적 제안 원칙과 일치합니다.

## 섹션 6: 청사진에서 현실로: 단계별 구현 로드맵

이 마지막 섹션에서는 애플리케이션을 구축하기 위한 전략적이고 단계적인 계획을 제공하여, 가치가 점진적으로 전달되고 가장 복잡한 기술적 과제가 조기에 위험 완화되도록 보장합니다.

### 6.1 1단계: 핵심 루프 (MVP)

* **초점:** 수동 상호작용을 통한 기본적인 "포착, 연결, 구체화" 루프를 구축합니다.
* **과업:**
  1. 섹션 3.1의 분석에 기반하여 플랫폼 결정(네이티브 vs. 리액트 네이티브)을 최종 확정합니다.
  2. 온디바이스 벡터 데이터베이스를 구현합니다 (섹션 2.2).
  3. 핵심 텍스트 및 음성 포착 UI를 구현합니다 (섹션 2.1).
  4. 선택된 온디바이스 임베딩 모델을 통합합니다 (섹션 3.2). 모든 새 항목은 임베딩되어 저장됩니다.
  5. 벡터 검색을 수행하고 결과를 표시하는 간단한 검색 인터페이스를 구축합니다.
  6. 온디바이스 LLM을 통합하고 RAG 파이프라인을 구현하여 사용자가 자신의 데이터에 대해 질문할 수 있도록 합니다 (섹션 3.3).
* **목표:** 전체 온디바이스 AI 파이프라인의 실행 가능성을 증명하는 기능적인 앱.

### 6.2 2단계: 지능 및 통합 강화

* **초점:** 통찰력의 질을 향상시키고 데이터 포착을 더 원활하게 만듭니다.
* **과업:**
  1. 하이브리드 검색(BM25 + 벡터) 및 RRF 순위 파이프라인을 구현합니다 (섹션 4.1 & 4.2).
  2. ReAct와 같은 고급 프롬프팅 기술을 구현합니다 (섹션 4.3).
  3. 시스템 전반의 공유 통합(iOS 공유 확장, 안드로이드 ACTION\_SEND)을 구축합니다 (섹션 2.1).
* **목표:** 훨씬 더 나은 통찰력을 생성하고 OS에 깊이 통합된 앱.

### 6.3 3단계: 앰비언트 컴패니언

* **초점:** 선제적이고 상황 인식적인 비전을 현실로 만듭니다.
* **과업:**
  1. 네이티브 컨텍스트 감지 API(Core Motion/Location, Awareness API)를 통합합니다 (섹션 5.1).
  2. 트리거 로직 및 규칙 엔진을 구축합니다 (섹션 5.2).
  3. 선제적 통찰력을 위한 고급, 비간섭적 UI 표면(동적 위젯 등)을 개발합니다 (섹션 2.3).
* **목표:** 사용자의 단편화된 자아를 연결하는 앰비언트, 선제적 동반자로서의 'Continuum' 비전의 완전한 실현.

## 결론

이 보고서는 'Continuum'이라는 사용자의 비전을 실행 가능한 기술 청사진으로 변환하기 위한 포괄적인 전략을 제시했습니다. 핵심 철학은 모든 사용자 데이터를 기기 내에서 처리하는 '설계 기반 프라이버시' 원칙에 기반한 완전한 온디바이스 아키텍처를 구축하는 것입니다. 이는 성능, 개인 정보 보호, 그리고 네트워크 독립성 측면에서 상당한 이점을 제공합니다.

주요 기술적 권장 사항은 다음과 같습니다.

1. **네이티브 개발 우선:** AI 처리, 백그라운드 작업, 최신 OS 기능과의 긴밀한 통합이라는 앱의 핵심 요구 사항을 고려할 때, Swift(iOS) 및 Kotlin(Android)을 사용한 네이티브 개발이 리액트 네이티브보다 장기적으로 더 강력하고 안정적인 기반을 제공할 것입니다.
2. **강력한 온디바이스 데이터베이스 채택:** ObjectBox와 같은 성숙한 온디바이스 벡터 데이터베이스는 빠른 시맨틱 검색과 효율적인 데이터 관리를 위한 견고한 기반을 제공합니다.
3. **고급 검색 파이프라인 구현:** 단순한 벡터 검색을 넘어, 키워드 기반 BM25 검색을 결합한 하이브리드 접근 방식과 상호 순위 융합(RRF)을 통한 지능형 결과 병합은 생성될 통찰력의 질을 극적으로 향상시킬 것입니다.
4. **효율적인 컨텍스트 감지 활용:** iOS의 Core Motion/Location 및 Android의 Awareness API와 같은 OS 네이티브, 저전력 API를 활용하여 사용자의 상황을 선제적으로 감지하고 관련 정보를 적시에 제공함으로써 진정한 '앰비언트' 경험을 실현할 수 있습니다.

제시된 3단계 구현 로드맵은 가장 중요한 기술적 위험을 초기에 해결하면서 점진적으로 가치를 구축하는 실용적인 경로를 제공합니다. MVP 단계에서 온디바이스 AI 파이프라인 전체의 실행 가능성을 검증하는 데 집중하고, 후속 단계에서 지능을 고도화하고 선제적 기능을 추가함으로써, 'Continuum'의 비전은 체계적이고 관리 가능한 방식으로 현실화될 수 있습니다. 이 청사진은 단순한 노트 앱을 넘어, 사용자가 자신의 삶의 서사를 재구성하도록 돕는 진정으로 개인적이고 지능적인 동반자를 만드는 길을 제시합니다.

#### 참고 자료

1. Google AI Edge - Gemini API, 8월 19, 2025에 액세스, <https://ai.google.dev/edge>
2. Mobile On-device AI - Reflections, 8월 19, 2025에 액세스, <https://annjose.com/post/mobile-ondevice-ai/>
3. On-Device AI Benchmarks: Nexa AI's Generative Model Performance Across Devices, 8월 19, 2025에 액세스, <https://nexa.ai/blogs/on-device-generative-ai-benchmarks>
4. AI on Android | Android Developers, 8월 19, 2025에 액세스, <https://developer.android.com/ai>
5. Want to Run LLMs on Your Device? Meet MLC, 8월 19, 2025에 액세스, <https://www.callstack.com/blog/want-to-run-llms-on-your-device-meet-mlc>
6. MLC LLM | Home, 8월 19, 2025에 액세스, <https://llm.mlc.ai/>
7. Live Transcribe | Speech to Text App - Android, 8월 19, 2025에 액세스, <https://www.android.com/accessibility/live-transcribe/>
8. Speech-to-Text AI: speech recognition and transcription - Google Cloud, 8월 19, 2025에 액세스, <https://cloud.google.com/speech-to-text>
9. Transferring large files with Share Extension : r/iOSProgramming - Reddit, 8월 19, 2025에 액세스, <https://www.reddit.com/r/iOSProgramming/comments/t8lsx0/transferring_large_files_with_share_extension/>
10. Reaching memory limit when sharing media assets from share extension to app via app group : r/iOSProgramming - Reddit, 8월 19, 2025에 액세스, <https://www.reddit.com/r/iOSProgramming/comments/hfk3vr/reaching_memory_limit_when_sharing_media_assets/>
11. Supporting iOS Share Extensions & Android Intents on React Native, 8월 19, 2025에 액세스, <https://www.devas.life/supporting-ios-share-extensions-android-intents-on-react-native/>
12. Intents and intent filters | App architecture - Android Developers, 8월 19, 2025에 액세스, <https://developer.android.com/guide/components/intents-filters>
13. Tasker 'Send Intent' action - Help/Guide - Google Groups, 8월 19, 2025에 액세스, <https://groups.google.com/g/tasker/c/clg8kEaaPdo>
14. Send simple data to other apps | App data and files - Android Developers, 8월 19, 2025에 액세스, <https://developer.android.com/training/sharing/send>
15. On-device vector database for Android (Java, Kotlin) - ObjectBox, 8월 19, 2025에 액세스, <https://objectbox.io/the-on-device-vector-database-for-android-and-java/>
16. The Swift database for iOS apps - ObjectBox, 8월 19, 2025에 액세스, <https://objectbox.io/swift-database-for-ios/>
17. On-Device Vector Search - ObjectBox Docs, 8월 19, 2025에 액세스, <https://docs.objectbox.io/on-device-vector-search>
18. ObjectBox, the edge vector database, 8월 19, 2025에 액세스, <https://objectbox.io/>
19. First on-device Vector Database (aka Semantic Index) for iOS - ObjectBox, 8월 19, 2025에 액세스, <https://objectbox.io/swift-ios-on-device-vector-database-aka-semantic-index/>
20. Tutorial Archives - ObjectBox, 8월 19, 2025에 액세스, <https://objectbox.io/category/tutorial/>
21. Kotlin Support - ObjectBox Docs, 8월 19, 2025에 액세스, <https://docs.objectbox.io/kotlin-support>
22. Tutorial: Demo Project - ObjectBox Docs, 8월 19, 2025에 액세스, <https://docs.objectbox.io/tutorial-demo-project>
23. Vector search: making sense of search queries - ObjectBox, 8월 19, 2025에 액세스, <https://objectbox.io/vector-search-making-sense-of-search-queries/>
24. Swift Archives - ObjectBox, 8월 19, 2025에 액세스, <https://objectbox.io/category/mobile-database/swift/>
25. Qdrant Edge: Vector Search for Embedded AI, 8월 19, 2025에 액세스, <https://qdrant.tech/blog/qdrant-edge/>
26. milvus-io/milvus-lite: A lightweight version of Milvus - GitHub, 8월 19, 2025에 액세스, <https://github.com/milvus-io/milvus-lite>
27. Milvus is a high-performance, cloud-native vector database built for scalable vector ANN search - GitHub, 8월 19, 2025에 액세스, <https://github.com/milvus-io/milvus>
28. Learn How to Use Chroma DB: A Step-by-Step Guide | DataCamp, 8월 19, 2025에 액세스, <https://www.datacamp.com/tutorial/chromadb-tutorial-step-by-step-guide>
29. Getting Started - Chroma Docs, 8월 19, 2025에 액세스, <https://docs.trychroma.com/getting-started>
30. Chroma, 8월 19, 2025에 액세스, <https://www.trychroma.com/>
31. MobileRAG: A Fast, Memory-Efficient, and Energy-Efficient Method for On-Device RAG - arXiv, 8월 19, 2025에 액세스, <https://arxiv.org/html/2507.01079v1>
32. objectbox/objectbox-java: Database for Android and JVM - first and fast, lightweight on-device vector database - GitHub, 8월 19, 2025에 액세스, <https://github.com/objectbox/objectbox-java>
33. What's new in iOS 18 - Apple Support (MN), 8월 19, 2025에 액세스, <https://support.apple.com/en-mn/guide/iphone/iphfed2c4091/ios>
34. What's new in iOS 18 - Apple Support, 8월 19, 2025에 액세스, <https://support.apple.com/guide/iphone/whats-new-in-ios-18-iphfed2c4091/ios>
35. Create a widget with Glance - Android Developers, 8월 19, 2025에 액세스, <https://developer.android.com/codelabs/glance>
36. Manage and update GlanceAppWidget | Jetpack Compose - Android Developers, 8월 19, 2025에 액세스, <https://developer.android.com/develop/ui/compose/glance/glance-app-widget>
37. The 9 best Android 15 features every smartphone owner should try, 8월 19, 2025에 액세스, <https://www.androidpolice.com/best-android-15-features/>
38. objective c - iOs Widget background update - Stack Overflow, 8월 19, 2025에 액세스, <https://stackoverflow.com/questions/28604185/ios-widget-background-update>
39. UI Design Patterns: Innovative Ways to Highlight Elements in Your App - Chameleon.io, 8월 19, 2025에 액세스, <https://www.chameleon.io/blog/new-design-patterns-highlighting-elements>
40. Notification UX: How To Design For A Better Experience - Userpilot, 8월 19, 2025에 액세스, <https://userpilot.com/blog/notification-ux/>
41. Design Guidelines For Better Notifications UX - Smashing Magazine, 8월 19, 2025에 액세스, <https://www.smashingmagazine.com/2025/07/design-guidelines-better-notifications-ux/>
42. Add support for the predictive back gesture | App architecture - Android Developers, 8월 19, 2025에 액세스, <https://developer.android.com/guide/navigation/custom-back/predictive-back-gesture>
43. Set up Predictive back | Jetpack Compose - Android Developers, 8월 19, 2025에 액세스, <https://developer.android.com/develop/ui/compose/system/predictive-back-setup>
44. Kotlin vs Swift: Which Language Should You Use for Native Mobile Development? - WEZOM, 8월 19, 2025에 액세스, <https://wezom.com/blog/kotlin-vs-swift-which-language-should-you-use-for-native-mobile-development>
45. Kotlin vs. Swift: What's the Difference? - Coursera, 8월 19, 2025에 액세스, <https://www.coursera.org/articles/kotlin-vs-swift>
46. React Native vs Native: The Ultimate Comparison, Which One is Better? (2025 Edition), 8월 19, 2025에 액세스, <https://www.iteratorshq.com/blog/react-native-vs-native-the-ultimate-comparison-which-one-is-better/>
47. Flutter vs React Native: Complete 2025 Framework Comparison Guide | Blog, 8월 19, 2025에 액세스, <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-comparison>
48. keywords:mediapipe - npm search, 8월 19, 2025에 액세스, <https://www.npmjs.com/search?q=keywords:mediapipe>
49. ThinkSys MediaPipe Package for React Native: iOS and Android, 8월 19, 2025에 액세스, <https://thinksys.com/development/thinksys-mediapipe-package-react-native/>
50. A React Native Camera and Player for MediaPipe applications. - GitHub, 8월 19, 2025에 액세스, <https://github.com/cdiddy77/react-native-mediapipe>
51. Integrating MediaPipe Hand Detection in React Native | by John Nguyen | Medium, 8월 19, 2025에 액세스, <https://medium.com/@dev-john-nguyen/integrating-mediapipe-hand-detection-in-react-native-f57c46aaab3a>
52. jigsawxyz/react-native-coreml-image: ReactNative CoreML Live Image Recognition through Machine Learning - GitHub, 8월 19, 2025에 액세스, <https://github.com/jigsawxyz/react-native-coreml-image>
53. AI-Powered React Native: Build Smarter Apps with Image Recognition, NLP and Predictive ML - Avidclan Technologies, 8월 19, 2025에 액세스, <https://www.avidclan.com/blog/ai-powered-react-native-build-smarter-apps-with-image-recognition-nlp-and-predictive-ml>
54. How to Build an AI-Based Mobile App with React Native - Solvios Technology, 8월 19, 2025에 액세스, <https://solvios.technology/how-to-build-ai-based-mobile-app-with-react-native/>
55. Handling location updates in the background | Apple Developer Documentation, 8월 19, 2025에 액세스, <https://developer.apple.com/documentation/corelocation/handling-location-updates-in-the-background>
56. Text embedding guide | Google AI Edge - Gemini API, 8월 19, 2025에 액세스, <https://ai.google.dev/edge/mediapipe/solutions/text/text_embedder>
57. sentence-transformers/all-MiniLM-L6-v2 - Hugging Face, 8월 19, 2025에 액세스, <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>
58. Speeding up Inference — Sentence Transformers documentation, 8월 19, 2025에 액세스, <https://sbert.net/docs/sentence_transformer/usage/efficiency.html>
59. Tune Mobile Performance (ORT <1.10 only) | onnxruntime, 8월 19, 2025에 액세스, <https://onnxruntime.ai/docs/performance/mobile-performance-tuning.html>
60. From Python To Android: HF Sentence Transformers (Embeddings) | by Shubham Panchal | ProAndroidDev, 8월 19, 2025에 액세스, <https://proandroiddev.com/from-python-to-android-hf-sentence-transformers-embeddings-1ecea0ce94d8>
61. Dripfarm/SVDB: Swift Vector Database. On-device, local vector database for building the next-generation of user experiences - GitHub, 8월 19, 2025에 액세스, <https://github.com/Dripfarm/SVDB>
62. Pooled word embeddings vs. sentence embeddings for short multilingual titles: Insights needed : r/LanguageTechnology - Reddit, 8월 19, 2025에 액세스, <https://www.reddit.com/r/LanguageTechnology/comments/17eg7cm/pooled_word_embeddings_vs_sentence_embeddings_for/>
63. The State of Embedding Technologies for Large Language Models — Trends, Taxonomies, Benchmarks, and Future Directions. | by Adnan Masood, PhD. | Medium, 8월 19, 2025에 액세스, <https://medium.com/@adnanmasood/the-state-of-embedding-technologies-for-large-language-models-trends-taxonomies-benchmarks-and-95e5ec303f67>
64. Choosing an Embedding Model - Pinecone, 8월 19, 2025에 액세스, <https://www.pinecone.io/learn/series/rag/embedding-models-rundown/>
65. LLM Inference guide for Android | Google AI Edge - Gemini API, 8월 19, 2025에 액세스, <https://ai.google.dev/edge/mediapipe/solutions/genai/llm_inference/android>
66. Everything you need to know about Google's new Gemma 7B and 2B Models - Medium, 8월 19, 2025에 액세스, <https://medium.com/@zaiinn440/everything-you-need-to-know-about-googles-new-gemma-7b-and-2b-models-5714dfaa48d7>
67. Best Offline LLMs for CPU | Gemma 2B vs Mistral vs Phi-2 vs Phi-3 Mini (Ollama Tested!), 8월 19, 2025에 액세스, <https://www.youtube.com/watch?v=4SdrvaIfFDM>
68. LLM Inference on Edge: A Fun and Easy Guide to run LLMs via React Native on your Phone!, 8월 19, 2025에 액세스, <https://huggingface.co/blog/llm-inference-on-edge>
69. Google's new MediaPipe LLM Inference API enables select large language models to run fully on-device. Supported models include Gemma 2B, Falcon 1B, Phi 2, and Stable LM 3B, while supported platforms are the web, Android, and iOS. : r/LocalLLaMA - Reddit, 8월 19, 2025에 액세스, <https://www.reddit.com/r/LocalLLaMA/comments/1ba5jeh/googles_new_mediapipe_llm_inference_api_enables/>
70. Quick Start — mlc-llm 0.1.0 documentation, 8월 19, 2025에 액세스, <https://llm.mlc.ai/docs/get_started/quick_start>
71. Top 10 Lightweight ML Frameworks for Edge and Mobile Devices in 2025 - Medium, 8월 19, 2025에 액세스, <https://medium.com/@eitbiz/top-10-lightweight-ml-frameworks-for-edge-and-mobile-devices-in-2025-fefc1b8d7d05>
72. What is Retrieval-Augmented Generation (RAG)? - Google Cloud, 8월 19, 2025에 액세스, <https://cloud.google.com/use-cases/retrieval-augmented-generation>
73. What is RAG? - Retrieval-Augmented Generation AI Explained - AWS, 8월 19, 2025에 액세스, <https://aws.amazon.com/what-is/retrieval-augmented-generation/>
74. AI Edge RAG guide - Gemini API, 8월 19, 2025에 액세스, <https://ai.google.dev/edge/mediapipe/solutions/genai/rag>
75. AI RAG with LlamaIndex, Local Embedding, and Ollama Llama 3.1 8b | by Michael Ruminer, 8월 19, 2025에 액세스, <https://m-ruminer.medium.com/ai-rag-with-llamaindex-local-embedding-and-ollama-llama-3-1-8b-b0620116a715>
76. Build RAG Application Using a LLM Running on Local Computer with Ollama Llama2 and LlamaIndex | by (λx.x)eranga | Effectz.AI | Medium, 8월 19, 2025에 액세스, <https://medium.com/rahasak/build-rag-application-using-a-llm-running-on-local-computer-with-ollama-and-llamaindex-97703153db20>
77. Using LLMs - LlamaIndex, 8월 19, 2025에 액세스, <https://docs.llamaindex.ai/en/stable/understanding/using_llms/using_llms/>
78. Starter Tutorial (Using Local LLMs) - LlamaIndex, 8월 19, 2025에 액세스, <https://docs.llamaindex.ai/en/stable/getting_started/starter_example_local/>
79. RAG (Retrieval Augmented Generation) with LlamaIndex, Elasticsearch and Mistral, 8월 19, 2025에 액세스, <https://www.elastic.co/search-labs/blog/rag-with-llamaIndex-and-elasticsearch>
80. RAG using Gemma, Langchain and ChromaDB - Kaggle, 8월 19, 2025에 액세스, <https://www.kaggle.com/code/gpreda/rag-using-gemma-langchain-and-chromadb>
81. Building a Multi-Modal RAG Pipeline with Langchain - Analytics Vidhya, 8월 19, 2025에 액세스, <https://www.analyticsvidhya.com/blog/2023/12/multi-modal-rag-pipeline-with-langchain/>
82. Implementing RAG using ‪@LangChain‬ and ChromaDB. Chat with your emails with this pipeline! - YouTube, 8월 19, 2025에 액세스, <https://www.youtube.com/watch?v=WcgoYKLkQos>
83. About hybrid search | Vertex AI | Google Cloud, 8월 19, 2025에 액세스, <https://cloud.google.com/vertex-ai/docs/vector-search/about-hybrid-search>
84. Knowledge Bases for Amazon Bedrock now supports hybrid search - AWS, 8월 19, 2025에 액세스, <https://aws.amazon.com/about-aws/whats-new/2024/03/knowledge-bases-amazon-bedrock-hybrid-search/>
85. How Hybrid Search is Changing the Game for AI Agents - Geeky Gadgets, 8월 19, 2025에 액세스, <https://www.geeky-gadgets.com/hybrid-search-for-ai/>
86. dorianbrown/rank\_bm25: A Collection of BM25 Algorithms in Python - GitHub, 8월 19, 2025에 액세스, <https://github.com/dorianbrown/rank_bm25>
87. Ranking: BM25+ - NucuLabs, 8월 19, 2025에 액세스, <https://blog.nuculabs.dev/posts/2024/2024-06-03-dsa-ranking-bm25/>
88. Unlocking the Power of Hybrid Search - A Deep Dive into Weaviate's Fusion Algorithms, 8월 19, 2025에 액세스, <https://weaviate.io/blog/hybrid-search-fusion-algorithms>
89. Distribution-Based Score Fusion (DBSF), a new approach to Vector Search Ranking | by Michelangiolo Mazzeschi | Plain Simple Software | Medium, 8월 19, 2025에 액세스, <https://medium.com/plain-simple-software/distribution-based-score-fusion-dbsf-a-new-approach-to-vector-search-ranking-f87c37488b18>
90. Relevance scoring in hybrid search using Reciprocal Rank Fusion (RRF) - Microsoft Learn, 8월 19, 2025에 액세스, <https://learn.microsoft.com/en-us/azure/search/hybrid-search-ranking>
91. RAG — VII (Reranking with RRF). Reciprocal Rank Fusion (RRF) is a rank… - Medium, 8월 19, 2025에 액세스, <https://medium.com/@danushidk507/rag-vii-reranking-with-rrf-d8a13dba96de>
92. Reciprocal Rank Fusion (RRF) explained in 4 mins — How to score results form multiple retrieval methods in RAG | by Deval Shah | Medium, 8월 19, 2025에 액세스, <https://medium.com/@devalshah1619/mathematical-intuition-behind-reciprocal-rank-fusion-rrf-explained-in-2-mins-002df0cc5e2a>
93. Introducing reciprocal rank fusion for hybrid search - OpenSearch, 8월 19, 2025에 액세스, <https://opensearch.org/blog/introducing-reciprocal-rank-fusion-hybrid-search/>
94. Hybrid Search Using Reciprocal Rank Fusion in SQL - SingleStore, 8월 19, 2025에 액세스, <https://www.singlestore.com/blog/hybrid-search-using-reciprocal-rank-fusion-in-sql/>
95. Rank Fusion Algorithms - From Simple to Advanced - Krystian Safjan's Blog, 8월 19, 2025에 액세스, <https://safjan.com/Rank-fusion-algorithms-from-simple-to-advanced/>
96. AI Document Summarization - IBM, 8월 19, 2025에 액세스, <https://www.ibm.com/architectures/hybrid/genai-document-summarization>
97. Prompt Engineering Guide to Summarization - PromptLayer, 8월 19, 2025에 액세스, <https://blog.promptlayer.com/prompt-engineering-guide-to-summarization/>
98. Examples of Prompts | Prompt Engineering Guide, 8월 19, 2025에 액세스, <https://www.promptingguide.ai/introduction/examples>
99. Summarizing Long Documents - OpenAI Cookbook, 8월 19, 2025에 액세스, <https://cookbook.openai.com/examples/summarizing_long_documents>
100. I need a prompt for summarizing documents : r/ChatGPTPromptGenius - Reddit, 8월 19, 2025에 액세스, <https://www.reddit.com/r/ChatGPTPromptGenius/comments/1dw06yb/i_need_a_prompt_for_summarizing_documents/>
101. Do Multi-Document Summarization Models Synthesize? - MIT Press Direct, 8월 19, 2025에 액세스, <https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00687/124262/Do-Multi-Document-Summarization-Models-Synthesize>
102. ReAct Prompting | Phoenix - Arize AI, 8월 19, 2025에 액세스, <https://arize.com/docs/phoenix/cookbook/prompt-engineering/react-prompting>
103. ReAct - Prompt Engineering Guide, 8월 19, 2025에 액세스, <https://www.promptingguide.ai/techniques/react>
104. ReAct Prompting: How We Prompt for High-Quality Results from LLMs | Chatbots & Summarization | Width.ai, 8월 19, 2025에 액세스, <https://www.width.ai/post/react-prompting>
105. Implement ReAct Prompting to Solve Complex Problems - Relevance AI, 8월 19, 2025에 액세스, <https://relevanceai.com/prompt-engineering/implement-react-prompting-to-solve-complex-problems>
106. ReAct Prompting - Leveraging Reasoning With AI - Silicon Dales, 8월 19, 2025에 액세스, <https://silicondales.com/ai/react-prompting/>
107. ReAct Prompting: Revolutionizing AI Interactions | Blog - FabriXAI, 8월 19, 2025에 액세스, <https://www.fabrixai.com/blog/react-prompting-revolutionizing-ai-interactions>
108. Core Location | Apple Developer Documentation, 8월 19, 2025에 액세스, <https://developer.apple.com/documentation/corelocation>
109. Best practice on user locations in background at higher accuracy than "significant changes"? (CLLocationManager accuracy question from iOS 8 onward) : r/iOSProgramming - Reddit, 8월 19, 2025에 액세스, <https://www.reddit.com/r/iOSProgramming/comments/4cfkrd/best_practice_on_user_locations_in_background_at/>
110. CMMotionActivity | Apple Developer Documentation, 8월 19, 2025에 액세스, <https://developer.apple.com/documentation/coremotion/cmmotionactivity>
111. CM MotionActivity background event trigger - Stack Overflow, 8월 19, 2025에 액세스, <https://stackoverflow.com/questions/33846867/cm-motionactivity-background-event-trigger>
112. Background Tasks and iOS Memory Management Best Practices - MoldStud, 8월 19, 2025에 액세스, <https://moldstud.com/articles/p-the-impact-of-background-tasks-on-ios-memory-management-challenges-and-best-practices>
113. What's the Awareness API? - Google for Developers, 8월 19, 2025에 액세스, <https://developers.google.com/awareness/overview>
114. Awareness API - Google for Developers, 8월 19, 2025에 액세스, <https://developers.google.com/awareness>
115. Migrate to Google Play services location and context APIs - Android Developers, 8월 19, 2025에 액세스, <https://developer.android.com/develop/sensors-and-location/location/migration>
116. Fused Location Provider API - Google for Developers, 8월 19, 2025에 액세스, <https://developers.google.com/location-context/fused-location-provider>
117. Increasing App Usage with Suggestions Based on User Activities - Apple Developer, 8월 19, 2025에 액세스, <https://developer.apple.com/documentation/foundation/task_management/increasing_app_usage_with_suggestions_based_on_user_activities>
118. Understanding iOS Proactive Suggestions - SitePoint, 8월 19, 2025에 액세스, <https://www.sitepoint.com/understanding-ios-proactive-suggestions/>