

N8n ile RAG Uygulaması Hazırlama

Doküman İşleme ve Veri Erişiminde Modern Yaklaşım



RAG



N8n



Supabase

21 Mayıs 2025

Sunum 1/10

RAG ve Fine-Tune: Temel Kavramlar ve Karşılaştırma



RAG

Retrieval-Augmented Generation

- ✓ Mevcut bilgi tabanından bilgileri alır ve LLM çıktılarıyla zenginleştirir
- ✓ Dış kaynaklardan alınan bilgiler sayesinde güncel bilgiye erişim sağlar
- ✓ Model eğitimi gerektirmez, çalışma esnasında karar verir
- ✓ Uygulaması hızlı ve maliyet etkindir
- ✓ Veri tabanı güncellenirse yanıtlar otomatik güncellenir

Sorgular dokümanlardaki bilgilere dayalı olarak cevaplanır



Fine-Tune

Model İnce Ayarı

- ✓ Önceden eğitilmiş modeli özel veri setiyle yeniden eğitir
- ✓ Belirli bir görev için optimize edilmiş sonuçlar sunar
- ✓ Eğitim için yüksek miktarda veri ve hesaplama kaynağı gerektirir
- ✓ Bilgi güncelleme için yeniden eğitim zorunludur
- ✓ Bilgiler modelin ağırlıklarında saklıdır

VS

Model, eğitim verilerine dayalı içsel bilgisini kullanır

RAG'ın Avantajları: Neden Daha Efektif?



Maliyet Avantajı

- ✓ Fine-tune'a göre %60-80 daha düşük maliyet
- ✓ Model eğitimi gerektirmez
- ✓ Donanım yatırımı gerekmez



Güncellenebilirlik

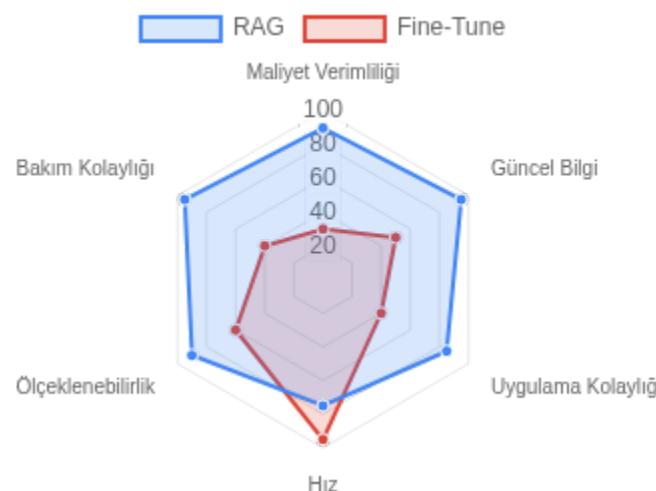
- ✓ Bilgi kaynakları anında güncellenebilir
- ✓ Yeni bilgiler hemen kullanılabilir
- ✓ Yeniden eğitim gerektirmez



Doğruluk ve Hassasiyet

- ✓ Doğrulanabilir dış kaynaklar kullanır
- ✓ Halüsinasyon riski daha düşük
- ✓ Kaynaklar referanslanabilir

RAG vs Fine-Tune: Kaynak Karşılaştırması



Kaynak Verimliliği

Daha az hesaplama gücü, hızlı cevap süresi



Hızlı Adaptasyon

Yeni senaryolara anında uyum sağlama



Veri Kontrolü

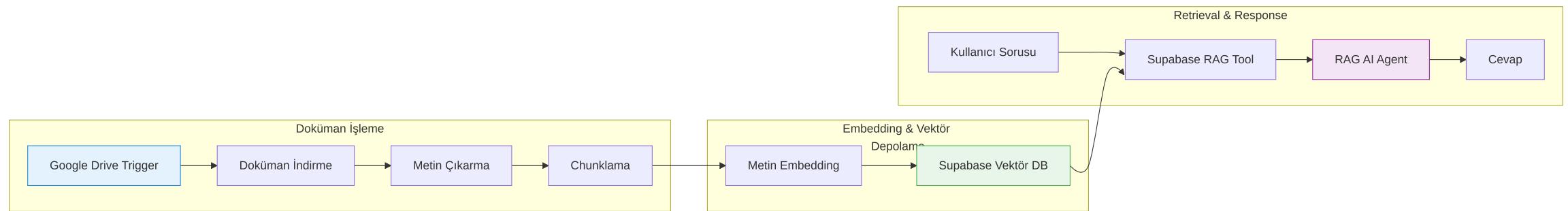
Hassas bilgileri modellerden izole etme



Ölçeklenebilirlik

Bilgi tabanı kolayca genişletilebilir

N8n ile RAG Mimarisi



Doküman İşleme

- ✓ Google Drive Belgelerini İzleme
- ✓ PDF, DOCX, TXT Formatlarını Destekler
- ✓ Recursive Character Text Splitter

Vektör Veritabanı

- ✓ Supabase pgvector Entegrasyonu
- ✓ Verimli Vektör Arama
- ✓ Metadata Filtreleme

Embeddings

- ✓ OpenAI Embeddings
- ✓ Ollama Yerel Alternatif
- ✓ Semantik Benzerlik Yakalama

Retrieval

- ✓ RAG AI Agent
- ✓ Postgres Chat Memory
- ✓ LLM Entegrasyonu

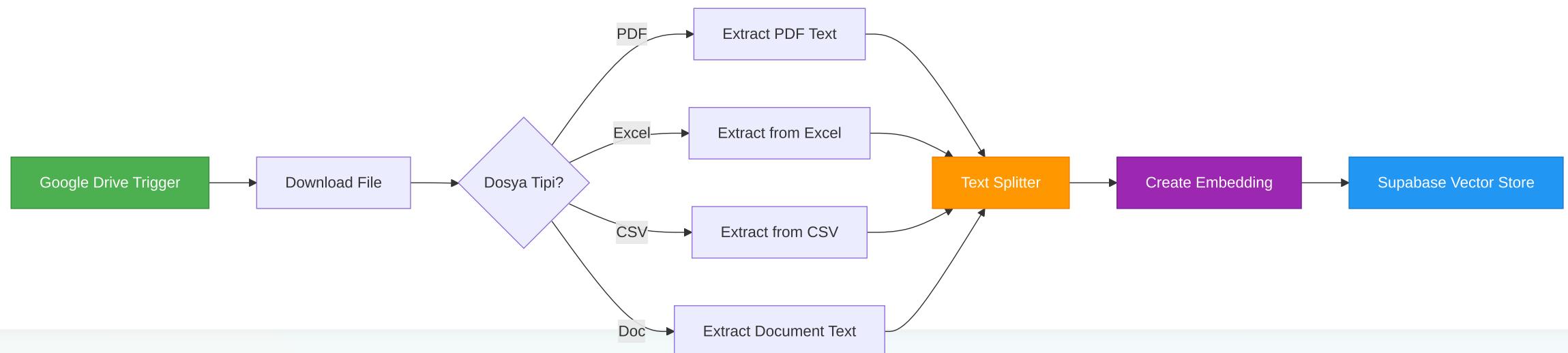
Veritabanı Sorusu

```
SELECT * FROM documents  
WHERE vector_column <-> $embedding < 0.3  
ORDER BY vector_column <-> $embedding  
LIMIT 5
```

N8n Workflow Avantajları

- ⚡ Kod yazmadan RAG sistemi kurulumu
- ⚡ Doküman ve soru arayüzü entegrasyonu
- ⚡ Farklı veri kaynaklarına kolay entegrasyon

Doküman İşleme Akışı (N8n Implementasyonu)



Google Drive Trigger

- ✓ Dosya oluşturma ve güncelleme tetikleyicileri
- ✓ Belirli klasörü periyodik olarak kontrol

Dosya İşleme

- ✓ Çoklu dosya formatı desteği (PDF, Excel, CSV, Doc)
- ✓ Format bazlı extractor node'ları
- ✓ Metin çıkarma ve birleştirme

```
// Switch node ile dosya tipine göre işleme
Switch ($json.file_type) {
  case "application/pdf":
    Extract PDF Text
  case "application/vnd.openxmlformats...":
    Extract from Excel
  case "application/vnd.google-apps.document":
    Extract Document Text
}
```

Text Chunking

- ✓ Recursive Character Text Splitter kullanımı
- ✓ Chunk boyutu: 400 karakter
- ✓ Overlap kontrollü parçalama

Embedding Oluşturma

- ✓ OpenAI Embeddings (text-embedding-3-small)
- ✓ Alternatif olarak Ollama (yerel model)

Vektör Veritabanı Saklama

- ✓ Supabase pgvector entegrasyonu
- ✓ Metadata bilgilerinin saklanması (file_id, file_title, file_url)
- ✓ Güncelleme durumunda eski verinin silinmesi

Önemli Noktalar

› Dosya güncellemeleri otomatik izlenir

› Her dosya güncelleme öncesi eski chunk'lar silinir

› Metadata ile kolay kaynak erişimi sağlanır

Retrieval İşlemi DATA ERİŞİM TEKNOLOJİSİ

OpenAI Chat Model

- ✓ Model: [gpt-4.1-mini](#)
- ✓ Sorguları anlama ve cevapları formatlama
- ✓ Dokümanlardan ilgili bilgi sentezleme

Document RAG Tool

- ✓ Vektör benzerlik araması
- ✓ Tablo: [documents](#)
- ✓ Sorğu: [match_documents](#)
- ✓ İlgili içerikleri bağlam olarak sağlama

Embeddings OpenAI

- ✓ Sorgu metinlerini vektöre çevirme
- ✓ Semantik anlam korunuğu
- ✓ Model: [text-embedding-3-small](#)
- ✓ 1536 boyutlu vektör oluşturma

Sorgu İşleme Akışı

1

Kullanıcı sorusu alınır

2

Sorgu vektöre çevrilir

3

Veritabanında aranır

4

İlgili dokümanlar bulunur

5

LLM cevap sentezler

Önemli Retrieval Özellikleri

- Semantik arama (anlam odaklı eşleştirme)
- Doküman chunk'larına hızlı erişim
- Cosine similarity ile vektör karşılaştırması
- Cevap kalitesi için kullanıcı sorğu optimizasyonu
- Metadata kullanarak kaynak bilgisi sunma
- Retrieval kalitesini artırmak için filtreler

RAG Ajansı

Postgres Chat Memory

- ✓ Kullanıcı konuşma geçmişini saklar
- ✓ Tablo: `memory_messages`
- ✓ Bağlam uzunluğu kontrolü
- ✓ JSON formatında mesaj geçmişi
- ✓ Oturum sürekliliği sağlar

RAG AI Agent Konfigürasyonu

- ✓ `promptType: define`
- ✓ Agent düşünme ve karar verme akışı
- ✓ Tool kullanım yeteneği (RAG & Integration)
- ✓ Detaylı sistem talimatları
- ✓ ReAct (Reason + Act) çerçevesini kullanır

Sistem Prompt

- ✓ Ajan rolünü ve sorumluluklarını tanımlar
- ✓ RAG için arama tekniklerini belirler
- ✓ Tutarlı ton ve yanıt formatı sağlar
- ✓ Hata yönetimi prosedürleri
- ✓ Tool kullanım stratejilerini açıklar

Sistem Prompt Örneği

```
## Role
You are a Retrieval-Augmented Generation (RAG) assistant designed to
You use a corpus of documents that are all text based.

## Responsibilities
- Answer user queries with a good mix of being comprehensive but still
- Retrieve and synthesize relevant information from the given tools
- Present information in an easy-to-understand manner
```

Ajan Karar Süreci

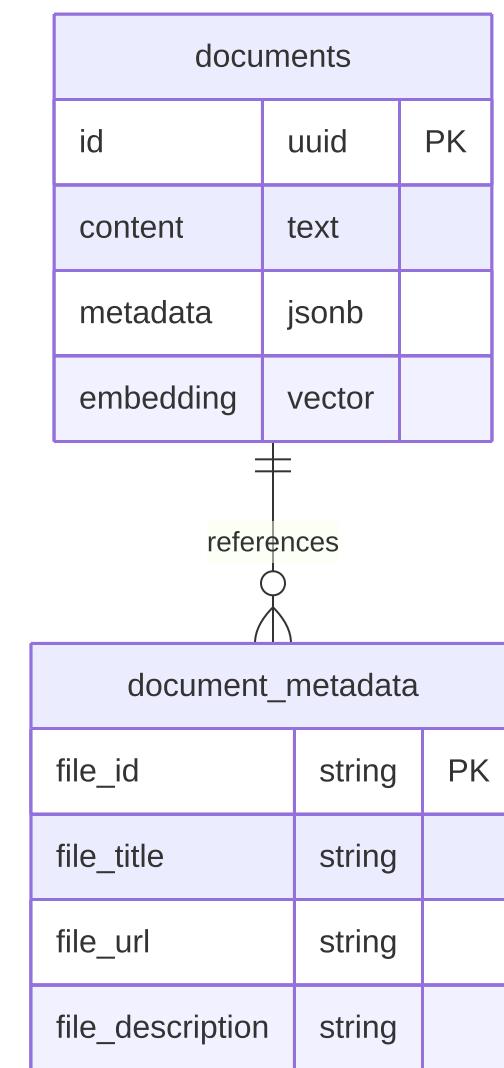
- 1 **Sorgu Anlama:** Kullanıcının sorusunu analiz eder ve niyeti belirler
- 2 **Tool Seçimi:** RAG için "documents" aracını kullanma kararı verir
- 3 **Hafıza Kontrolü:** Önceki konuşma bağlamını değerlendirir
- 4 **Veri Erişimi:** Doküman verilerine erişmek için RAG aracını kullanır
- 5 **Sentez ve Yanıt:** Bulunan bilgileri analiz edip cevap oluşturur

RAG Ajansının Temel Özellikleri

- Çoklu araç (tool) kullanabilme yeteneği
- Oturumlar arası hafıza sürekliliği
- Akıllı tool seçimi ve ne zaman kullanılacağını bilme
- İşlevsel yanıtlar için otomatik düşünme süreci
- Soruyu iyileştirme ve genişletme yeteneği
- Doğru ve alakalı bilgi çağrıma odaklı

Supabase Vektör Veritabanı

Veritabanı Yapısı PGVECTOR



</> Benzerlik Sorgusu SQL Örneği

```
-- Supabase'de Cosine Benzerlik Sorgusu
SELECT
    id,
    ...
    ...
Vektör arama için <=> operatörü cosine mesafesi hesaplar (1 - mesafe = benzerlik)
```

Vektör Depolama Özellikleri

- ✓ Vektör (pgvector) tipinde saklama ve arama
- ✓ Metadata JSON formatında doküman bilgisi
- ✓ OpenAI embeddings ile vektörizasyon
- ✓ Cosine similarity ile semantik arama
- ✓ PostgreSQL RLS ile güvenlik kontrolleri
- ✓ Doküman bilgilerine hızlı erişim ve filtreleme

Vectorstore Node Konfigürasyonu

Vektör Ekleme (Insert Mode)

Mode: Table Name:

Vektör Sorgulama (Retrieve Mode)

Mode: Query:

İndeksleme ve Performans

- ✓ HNSW indeksleme ile yüksek performans
- ✓ Vektör boyutu: 1536 (OpenAI text-embedding-3-small)
- ✓ Index oluşturma: CREATE INDEX ON documents USING ivfflat (embedding)
- ✓ Otomatik chunklama ve vektöre dönüştürme

Dikkat Edilecek Noktalar

- Vektör boyutları embedding modeline göre ayarlanmalı
- Büyük veri setleri için index optimizasyonu yapılmalıdır
- Benzerlik eşiği (threshold) veri setine göre kalibre edilmeli

N8n RAG Sistemi Entegrasyon Noktaları

Vektör Veritabanı İşlemleri
CRUD operasyonları ve semantik arama

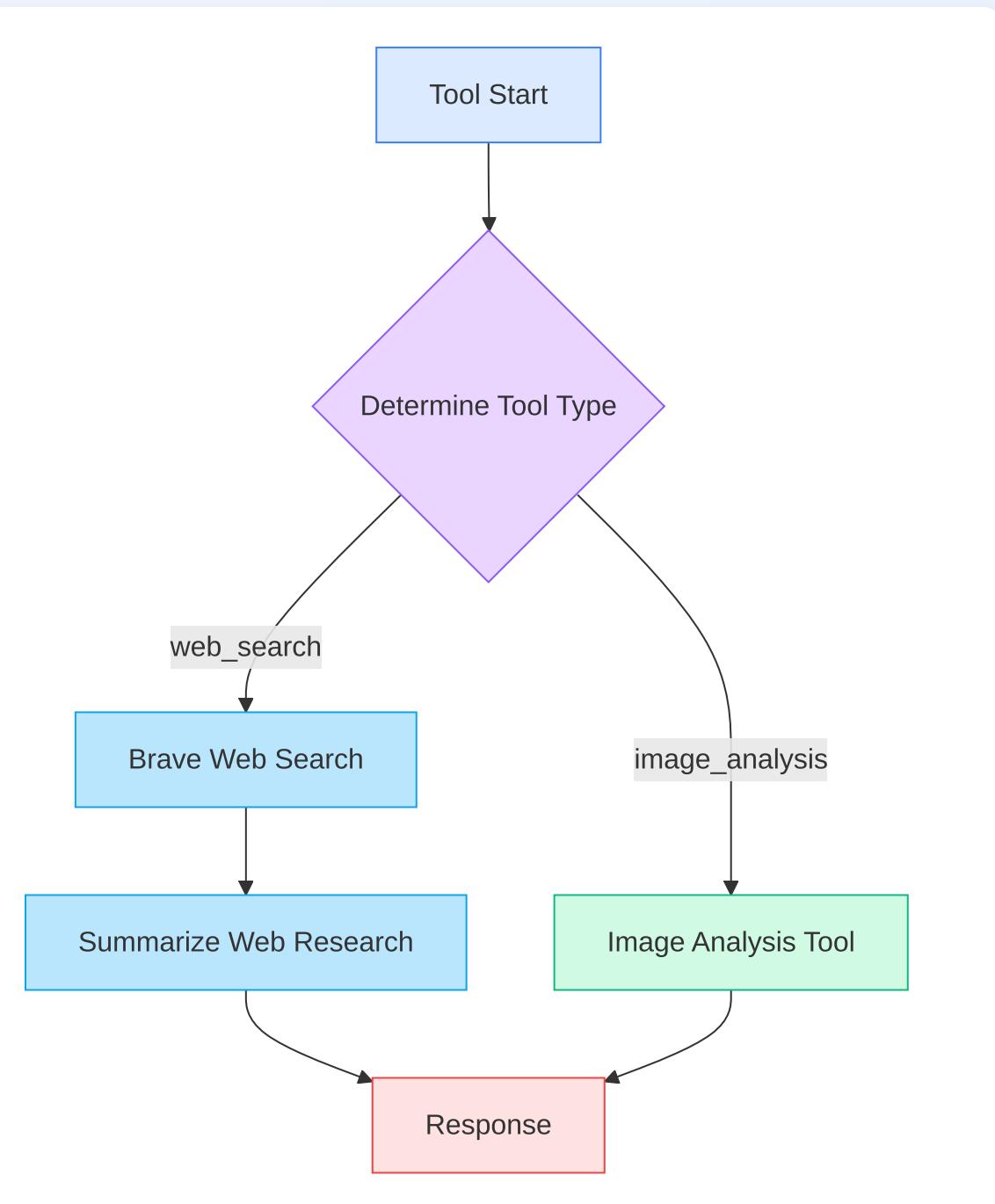
Metadata İşleme
JSON formatında metadata saklama ve filtreleme

LLM Entegrasyonu
AI Tool olarak RAG işlemlerini gerçekleştirme

Veri Yönetimi
Otomatik veri güncelleme ve temizleme

Internet Search Entegrasyonu

Q Internet Search İş Akışı N8N WORKFLOW



Brave Search API Entegrasyonu

- ✓ Gerçek zamanlı web araması yapma
- ✓ Arama sonuçlarını özetleme
- ✓ RAG veritabanında olmayan bilgileri tamamlama
- ✓ Tool olarak RAG ajanından çağrılabilme
- ✓ API isteklerini özelleştirebilme

⚙️ Tool Türleri ve Konfigürasyon

Web Search Tool

Tool Type: `web_search`
Query Parametresi: `query`

Image Analysis Tool

Tool Type: `image_analysis`
URL Parametresi: `image_url`

</> API İstek Örneği

```
// Brave Web Search API Request
GET https://api.search.brave.com/res/v1/web/search?q={{ query }}&summ
Headers:
Accept: application/json
Accept-Encoding: gzip
Authorization: Bearer [API KEY]
```

Sorgu sonuçları öznetmek için `summarizer.key` parametresi kullanılır

🔌 RAG ile Entegrasyon

Dış Kaynak Zenginleştirme

Internet araması ile RAG sistemini gerçek zamanlı bilgilerle güçlendirme

Hibrit Retrieval

Hem veritabanından hem de web kaynaklarından bilgi alma ve birleştirme

Etki Alanı Genişletme

RAG sisteminin kapsama alanını genişletecek daha geniş sorulara cevap verebilme

Dikkat Edilecek Noktalar

- API istek limitleri ve kota yönetimi
- Internet aramalarında yanıt süreleri optimize edilmeli
- Sensitive veya özel bilgilerin aranmaması için filtre mekanizması gereklidir

🔧 Internet Araması Temel Bileşenler

Tool Başlatma

İş akışını tetikleyen özel parametreler

Tool Türü Belirleme

Gelen isteğin hangi araç tipine yönlendirileceği

Web Arama API

Brave Search API ile sorguları işleme

Özetleme

Arama sonuçlarını özet formata dönüştürme

💡 Önemli Noktalar

- ✓ Internet search tool'u RAG ajanının dış dünyaya bağlantısını sağlar
- ✓ Tool tipi belirlenerek uygun API isteği gerçekleştirilebilir

⚙️ Kullanım Alanları

- ✓ Güncel bilgilerin entegrasyonu (haberler, güncel veriler)
- ✓ Yetersiz kaldığı konularda RAG sistemini tamamlama

N8n Node Konfigürasyon Detayları

OpenAI Chat Model

LLM

Model: `gpt-4.1-mini` (daha iyi performans için `gpt-4-turbo`)

Temperature: `0.7` (0.0 daha belirleyici yanıtlar için)

Max Tokens: `4000` (Yanıt uzunluğu)

```
"parameters": { "model": "gpt-4.1-mini", "options": { "temperature": 0.7, "maxTokens": 4000 } }
```

İpuçu: Sistem mesajı ile LLM'in rolünü ve davranışını özelleştirebilirsiniz.

Embeddings OpenAI

VEKTÖRLEŞTİRME

Model: `text-embedding-3-small` (Uygun maliyet ve iyi performans)

Boyut: `1536` (Çıktı vektör boyutu)

Alternatif: `Ollama Embeddings` (Yerel çalışma için)

```
"parameters": { "model": "text-embedding-3-small", "options": {} }
```

Avantaj: Yüksek kaliteli vektör oluşturma
Semantik arama hassasiyeti

Alternatif: Ollama (yerel, ücretsiz)
Nomic-embed-text (açık kaynak)

Recursive Character Text Splitter

CHUNKLAMA

Chunk Size: `400` (Token limitine göre ayarlayın)

Chunk Overlap: `40-80` (Chunk Size'ın ~%10-20'si)

Separator: `\n\n` (Paragraf bazlı bölünme)

```
"parameters": { "chunkSize": 400, "chunkOverlap": 40, "options": {} }
```

Optimum Chunk Size:

- Çok küçük: Anlamlı bilgi eksikliği riski
- Çok büyük: Gürültü artışı ve ilgisiz içerik riski
- İdeal: 300-500 karakter arası

Supabase Vector Store

VERİ DEPOLAMA

Mode: `insert / retrieve-as-tool`

Table Name: `documents`

Query Name: `match_documents` (Cosine similarity sorgusu)

```
"parameters": { "mode": "retrieve-as-tool", "toolName": "documents", "toolDescription": "Use RAG to look up information in the knowledgebase.", "tableName": { "value": "documents" } }
```

Vektör Depolama:

- Vektör tipi: pgvector
- Benzerlik: cosine
- Match count: 3-5 arası

Metadata Yapısı:

- file_id: Doküman ID'si
- file_title: Doküman başlığı
- file_url: Kaynak URL

★ Optimum Konfigürasyon İpuçları

Performans Optimizasyonu

- Bellek yönetimi için verimli chunk boyutu
- Küçük embeddings modelleriyle başlayıp gerektiğinde büyük modeller
- Ollama ile yerel işleme seçeneği

Entegrasyon Noktaları

- Bellek (Postgres Chat Memory)
- Doc Vektörizasyonu (Embeddings)
- Agent Prompts (Sistem Mesajları)
- External Tools (Internet Search)

Hata Yönetimi

- Fallback cevaplar için sistem mesajı
- Vektör benzerlik eşliğini dinamik ayarlama
- Eski verileri temizleme (Delete Old Doc Rows)

Uygulama Örnekleri ve Sonuçlar



Dokümantasyon Arama

Kullanıcılar büyük teknik dokümantasyonlar içinde doğru bilgiyi hızlı bir şekilde bulabilir.

HIZLI YANIT

BAĞLAMSAL İÇERİK



Bilgi Tabanı Sorgulama

Şirket içi bilgi tabanlarını doğal dil sorguları ile arayarak kurumsal hafızayı etkin kullanma.

SEMANTİK ARAMA

ÇAPRAZ REFERANS



Dinamik İçerik Analizi

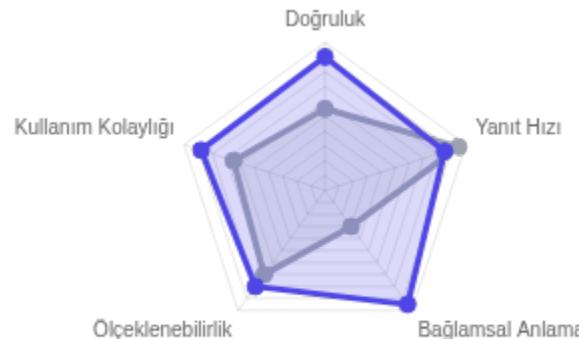
Sürekli güncellenen içeriklerin otomatik işlenerek bilgi tabanının güncellliğini koruması.

OTOMATİK

GERÇEK ZAMANLI

Performans Metrikleri

N8n RAG vs. Geleneksel Yöntemler



N8n RAG

Geleneksel Arama

● N8n RAG: Düşük latency, yüksek doğruluk

● SQL Sorgular: Orta hız, katı eşleşme

● Regex Arama: Yüksek hız, düşük hassasiyet

● RAG sisteminde hassasiyet: %91

Sorgu Karşılaştırması

"Finans raporunda Q2 sonuçları"

Basit Arama İlgisiz dökümanlar, yanlış eşleşme

"Finans raporunda Q2 sonuçları"

RAG Doğrudan Q2 finans verilerine odaklanma

"Proje detay analizi?"

Basit Arama Anlamlı sonuç yok veya genel bilgi

"Proje detay analizi?"

RAG Proje analiz dökümantasyonu özeti



Kullanıcı Geri Bildirimi

Teknik Ekip Lideri



"N8n ile oluşturduğumuz RAG sistemi sayesinde ekibimizin teknik doküman incelemesi süresi %70 azaldı. Artık sadece bilginin yerini değil, anlamını da bulmamızı sağlıyor."

Kapanış ve İletişim

Gelecek Geliştirmeler



Multi-modal RAG

Görsel, ses ve metin entegrasyonu ile çoklu format desteği



Streaming RAG

Gerçek zamanlı veri akışıyla anlık bilgi işleme ve yanıt



Çoklu Dil Desteği

Farklı dillerdeki dokümanlar için RAG entegrasyonu



Gelişmiş Güvenlik

Hassas verilerde daha güçlü erişim kontrolü ve şifreleme

Sonraki Adımlar

1

Performans Optimizasyonu

Büyük ölçekli verilerde chunk boyutu ve overlap parametrelerini optimize etme

2

Metadata Zenginleştirme

Daha hassas aramaları desteklemek için metadata yapısını geliştirme

3

Otomasyon Genişletme

Diğer doküman kaynaklarını (SharePoint, CRM sistemleri) entegre etme

Teşekkürler!

N8n ve RAG entegrasyonu ile ilgili bu sunum boyunca gösterdiğiniz ilgi için teşekkür ederiz. RAG teknolojisi, bilgi erişiminde yeni bir çağ başlatmakta ve N8n ile bu gücü kurumunuza entegre etmek artık çok daha kolay.

Kaynaklar ve Dökümantasyon:

- N8n Resmi Dökümantasyon
- Supabase Vektör Veritabanı Kılavuzu
- OpenAI API Referansı



Sorular?

Sistemin kurulumu, konfigürasyonu veya özel senaryolar hakkında sorularınız için hazırlız!