GraphRAG & n8n Setup - Cheat Sheet

Überblick

Dieses Cheat Sheet zeigt, wie du KI-Agenten mit GraphRAG und n8n zehnmal intelligenter machst. GraphRAG löst die Hauptprobleme traditioneller RAG-Systeme: Kontextverlust, fehlende Beziehungsverständnis und schlechte Antworten bei komplexen Fragen.

Was ist GraphRAG?

Wissensgraph Grundlagen

- **Knoten/Entitäten**: Personen, Kurse, Unternehmen (dargestellt als Kreise)
- **Beziehungen**: Verbindungen wie "arbeitet bei", "lebt in", "unterrichtet"
- **Eigenschaften**: Zusätzliche Infos wie "Computerscience-Kurs", "englischsprachig"

GraphRAG vs. Traditionelles RAG

Traditionelles RAG:

- Sucht isolierte Textschnipsel
- Verliert Kontext zwischen Informationen
- Versteht keine Beziehungen

GraphRAG:

- Nutzt Wissensgraphen für Zusammenhänge
- Kombiniert Vektorspeicher mit Graph-Datenbank
- Liefert kontextgenaue, fundierte Antworten

LightRAG vs. Microsoft GraphRAG

Microsoft GraphRAG:

- Sehr leistungsfähig bei globalen Fragen
- Teuer, langsam, schwer zu aktualisieren

LightRAG:

- Schlankere Alternative
- Dual-Level Retrieval (lokal + global)
- Günstiger, schneller, einfacher
- Open-Source

LightRAG Cloud-Setup

Voraussetzungen

- Render.com Account
- OpenAI API Key

Render.com Deployment

1. **Neues Projekt erstellen**
    - Gehe zu Render.com
    - Klicke auf "New" → "Web Service"
    - Wähle "Deploy an existing image"
2. **Docker Image konfigurieren**
    - Image URL: `ghcr.io/hkuds/lightrag:latest`
    - Name: Wähle einen Namen für deinen Service
    - Region: Wähle die nächstgelegene Region
    - Branch: main (Standard)
3. **Instanztyp wählen**
    - Plan: Starter ($7/Monat) - ausreichend für die meisten Anwendungen
    - Auto-Deploy: Aktiviert lassen
4. **Persistente Speicherung**
    - Gehe zu "Advanced"
    - Klicke "Add Disk"
    - Name: lightrag-data
    - Mount Path: `/var/data`
    - Size: 1 GB (kann später erweitert werden)
5. **Deploy starten**
    - Klicke "Deploy"
    - Warte auf Deployment-Abschluss

Umgebungsvariablen Setup

Wichtig: Alle Umgebungsvariablen müssen vor dem ersten Deployment gesetzt werden.

1. Basis-Authentifizierung

AUTH_ACCOUNTS

- Wert: `nutzer123:passwort123`
- Beschreibung: Login-Daten für LightRAG-Interface
- Format: Benutzername:Passwort
- Beispiel: `admin:meinpasswort2024`

LIGHTRAG_API_KEY

- Wert: `apikey123`
- Beschreibung: API-Schlüssel für n8n-Integration
- Beispiel: `lightrag_key_2024_secure`
- **Merke dir diesen Key - du brauchst ihn später für n8n!**

2. OpenAI & Embedding-Konfiguration

Füge diese Variablen über "Add Environment Variable" hinzu:

Embedding-Konfiguration:

```
EMBEDDING_BINDING=openai
EMBEDDING_MODEL=text-embedding-3-small
EMBEDDING_DIM=1536
EMBEDDING_BINDING_HOST=https://api.openai.com/v1
EMBEDDING_BINDING_API_KEY=your_openai_api_key
```

LLM-Konfiguration:

```
LLM_BINDING=openai
   LLM_MODEL=gpt-4.1-nano
LLM_BINDING_HOST=https://api.openai.com/v1
LLM_BINDING_API_KEY=your_openai_api_key
```

Performance-Einstellungen:

```
MAX_ASYNC=6
MAX_PARALLEL_INSERT=3
EMBEDDING_FUNC_MAX_ASYNC=18
EMBEDDING_BATCH_NUM=100
```

3. Schritt-für-Schritt Anleitung

1. **OpenAI API Key besorgen**
   - Gehe zu platform.openai.com
   - Erstelle API Key
   - Ersetze `your_openai_api_key` mit deinem echten Key
2. **Variablen in Render eintragen**

   Einfache Methode - Alle auf einmal:

   - Klicke "Add from .env"
   - Füge folgenden kompletten Text ein:

```
AUTH_ACCOUNTS=admin:meinpasswort123
LIGHTRAG_API_KEY=lightrag_secure_key_2024
EMBEDDING_BINDING=openai
EMBEDDING_MODEL=text-embedding-3-small
EMBEDDING_DIM=1536
EMBEDDING_BINDING_HOST=https://api.openai.com/v1
EMBEDDING_BINDING_API_KEY=sk-proj-xxxxxxxxxxxxxxx
LLM_BINDING=openai
LLM_MODEL=gpt-4.1-nano
LLM_BINDING_HOST=https://api.openai.com/v1
LLM_BINDING_API_KEY=sk-proj-xxxxxxxxxxxxxxx
MAX_ASYNC=6
MAX_PARALLEL_INSERT=3
EMBEDDING_FUNC_MAX_ASYNC=18
EMBEDDING_BATCH_NUM=100
```

    - **Wichtig:** Ersetze `sk-proj-xxxxxxxxxxxxxxx` mit deinem echten OpenAI API Key
    - **Wichtig:** Ändere `admin:meinpasswort123` zu deinen gewünschten Login-Daten
    - **Wichtig:** Ändere `lightrag_secure_key_2024` zu deinem gewünschten API Key
    - Klicke "Add Variables"
3. **Login nach Deployment**

Nach erfolgreichem Deployment:

- URL: [Deine Render Service URL]
- Username: Was du bei AUTH_ACCOUNTS vor dem ":" eingetragen hast
- Passwort: Was du bei AUTH_ACCOUNTS nach dem ":" eingetragen hast

---

LightRAG Benutzeroberfläche

Hauptbereiche

Dokumentenbereich:

- Upload von Dokumenten
- Automatische Verarbeitung:
    1. Duplikate herausfiltern
    2. Text in Chunks aufteilen
    3. Vektoren erstellen und speichern
    4. Entitäten und Beziehungen extrahieren
    5. Konsolidierung bei 4+ Referenzen

Knowledge Graph:

- Visuelle Darstellung des Graphen
- Zoom- und Navigationsfunktionen
- Entitätseigenschaften anzeigen
- Beziehungen verfolgen

Retrieval-Tab:

- Testumgebung für Abfragen
- Verschiedene Modi verfügbar
- Kontext oder LLM-Antworten

Abfragemodi

- **Naive**: Einfache Textähnlichkeitssuche
- **Local**: Präzise Treffer im unmittelbaren Kontext
- **Global**: Semantische Suche im gesamten Graphen
- **Hybrid**: Kombination aus Vektor- und Graph-Suche
- **Mix**: Empfohlener Modus für beste Ergebnisse

N8N Integration

Workflow Import

Um die Übersichtlichkeit zu gewährleisten findest du den Workflow am Ende des Dokuments.

📥 Workflow Import Anleitung:

1. Kopiere das komplette JSON oben
2. Gehe in n8n zu "Templates" → "Import from JSON"
3. Füge das JSON ein und klicke "Import"
4. Der Workflow wird automatisch erstellt

🔧 Nach dem Import konfigurieren:

1. **OpenAI API Key hinzufügen:**
    - Gehe zu "Credentials" in n8n
    - Füge deine OpenAI API Credentials hinzu
    - Verknüpfe sie mit dem "OpenAI Chat Model" Node
2. **LightRAG API Key finden:**
    - Gehe in deine LightRAG-Anwendung
    - Klicke auf den "API"-Tab
    - Klicke auf "Authorize"
    - Unter "APIKey Header" gibst du deinen API Key an und wählst nochmal "Authorize"
    - **Das ist derselbe Key wie dein LIGHTRAG_API_KEY aus den Umgebungsvariablen**

HTTP Request Node Konfiguration

1. **cURL Import**
    - Gehe in LightRAG zu API → Post Query
    - Klicke "Execute" um Schema zu laden
    - Kopiere den generierten cURL-Befehl
    - In n8n: HTTP Request Node → "Import cURL"
2. **Authentifizierung Setup**
    - Authentication: Generic Credential Type
    - Auth Type: Header Auth
    - Erstelle neues Credential
    - Name: X-API-Key
    - Value: [LightRAG API Key aus API-Tab]
3. **Body Konfiguration**
    - Body Type: JSON
    - Specify Body: Using Fields below
    - Header Parameter: query
    - Value: Automatisch aus vorherigem Node

Agent Integration

- Tool wird automatisch ausgelöst bei relevanten Fragen
- LightRAG generiert detaillierte Antworten mit Quellenangaben
- Nahtlose Integration in bestehende n8n-Workflows

Troubleshooting

Häufige Probleme

Deployment Fehler:

- Überprüfe Docker Image URL
- Stelle sicher, dass alle Umgebungsvariablen gesetzt sind
- Kontrolliere Disk-Mount-Path

API Verbindungsfehler:

- Validiere API Key
- Prüfe Service-URL
- Überprüfe Header-Authentifizierung in n8n

Schlechte Antwortqualität:

- Lade mehr relevante Dokumente hoch
- Teste verschiedene Abfragemodi
- Überprüfe Token-Limits für Entitäten

Performance Optimierung

Token Management:

- Setze maximale Token-Größen für Entitäten
- Begrenze Beziehungs-Token
- Überwache API-Verbrauch

Skalierung:

- Erhöhe Disk-Größe bei mehr Dokumenten
- Upgrade Render-Plan bei höherem Traffic
- Nutze Parallelitäts-Einstellungen

Nächste Schritte

1. **Erste Dokumente hochladen und testen**
2. **N8N-Workflow an eigene Bedürfnisse anpassen**
3. **Monitoring und Optimierung einrichten**

Der Workflow:

```
{
  "name": "GraphRAG Agent Workflow",
  "nodes": [
    {
      "parameters": {
        "options": {}
      },
      "id": "51219c75-24d2-46de-bcdd-54c509cb8d67",
      "name": "When chat message received",
      "type": "@n8n/n8n-nodes-langchain.chatTrigger",
      "typeVersion": 1.1,
      "position": [
        320,
        120
      ],
      "webhookId": "graph-rag-chat"
    },
    {
      "parameters": {
        "options": {
          "systemMessage": "=Du bist ein Graph RAG Experte. Nutze IMMER das
Graph Knowledge Expert Tool für deine Antworten.",
          "returnIntermediateSteps": true
```

```
        }
      },
      "id": "13519b5f-12b4-437f-97d4-f56d622530ae",
      "name": "AI Agent",
      "type": "@n8n/n8n-nodes-langchain.agent",
      "typeVersion": 1.7,
      "position": [
        540,
        120
      ]
    },
    {
      "parameters": {
        "model": "gpt-4.1-nano",
        "options": {
          "frequencyPenalty": 0,
          "maxTokens": 4000,
          "responseFormat": "text",
          "presencePenalty": 0,
          "temperature": 0.1,
          "timeout": 60000,
          "maxRetries": 2,
          "topP": 0.9
        }
      },
      "id": "ce8ed2dd-e59d-4b2b-a277-7354180e4256",
      "name": "OpenAI Chat Model",
      "type": "@n8n/n8n-nodes-langchain.lmChatOpenAi",
      "typeVersion": 1.1,
      "position": [
        540,
        280
      ],
      "credentials": {
        "openAiApi": {
          "id": "DEINE-OPENAI-CREDENTIAL-ID",
          "name": "OpenAI Account"
        }
      }
    },
    {
      "parameters": {
        "sessionIdType": "customKey",
        "sessionKey": "={{ $('When chat message
received').item.json.sessionId }}",
        "contextWindowLength": 10
      },
      "id": "661902a1-dbc7-4f5d-896a-1792795a119d",
      "name": "Window Buffer Memory",
      "type": "@n8n/n8n-nodes-langchain.memoryBufferWindow",
      "typeVersion": 1.2,
      "position": [
        540,
        360
      ]
    },
    {
      "parameters": {},
      "id": "386aa5bb-2804-4744-861f-e9d45c5a067c",
      "name": "Graph Knowledge Expert",
      "type": "@n8n/n8n-nodes-langchain.toolHttpRequest",
      "typeVersion": 1.2,
      "position": [
```

```
        540,
        440
      ]
    },
    {
      "parameters": {
        "content": "## 🤖 Graph RAG Agent Setup\n\n### Wichtige Konfigurationsschritte:\n\n1. **LightRAG Service URL anpassen**\n   - Ersetze `YOUR-LIGHTRAG-SERVICE` mit deiner tatsächlichen Service-URL\n\n2. **API Credentials erstellen**\n   - Gehe zu Credentials → Add Credential\n - Wähle \"Header Auth\"\n   - Name: `lightragApi`\n   - Header Name: `X-API-Key`\n   - Header Value: Dein LightRAG API Key\n\n3. **OpenAI Credentials**\n   - Füge deine OpenAI API Credentials hinzu\n\n4. **Workflow aktivieren**\n   - Klicke auf \"Activate\" oben rechts\n\n### Test:\nÖffne den Chat und stelle Fragen wie:\n- \"Welche Beziehungen bestehen zwischen [Entität A] und [Entität B]?\"\n- \"Erkläre mir die Zusammenhänge im Dokument X\"\n- \"Was sind die wichtigsten Konzepte und wie hängen sie zusammen?\"",
        "height": 441.5293209876547,
        "width": 396.0802469135803,
        "color": 4
      },
      "id": "91b8495b-6597-402c-be53-827a2cdcc860",
      "name": "Sticky Note",
      "type": "n8n-nodes-base.stickyNote",
      "typeVersion": 1,
      "position": [
        -260,
        120
      ]
    },
    {
      "parameters": {
        "jsCode": "// Response Processing & Formatting\nconst response = $input.item.json;\n\n// Extrahiere strukturierte Daten aus der LightRAG Response\nlet processedResponse = {\n  query: response.query || '',\n  answer: '',\n  entities: [],\n  relationships: [],\n  sources: [],\n  confidence: 0,\n  mode: response.mode || 'hybrid'\n};\n\n// Verarbeite die Antwort\nif (response.response) {\n  processedResponse.answer = response.response;\n}\n\n// Extrahiere Entitäten (falls vorhanden)\nif (response.entities && Array.isArray(response.entities)) {\n  processedResponse.entities = response.entities.map(entity => ({\n    name: entity.name || entity.entity || '',\n    type: entity.type || 'unknown',\n    description: entity.description || '',\n    properties: entity.properties || {}\n  }));\n}\n\n// Extrahiere Beziehungen (falls vorhanden)\nif (response.relationships && Array.isArray(response.relationships)) {\n  processedResponse.relationships = response.relationships.map(rel => ({\n    source: rel.source || rel.from || '',\n    target: rel.target || rel.to || '',\n    type: rel.relationship || rel.type || '',\n    strength: rel.strength || rel.weight || 1,\n    description: rel.description || ''\n  }));\n}\n\n// Extrahiere Quellen\nif (response.sources && Array.isArray(response.sources)) {\n  processedResponse.sources = response.sources.map(source => ({\n    document: source.document || source.file || '',\n    page: source.page || null,\n    relevance: source.relevance || source.score || 0,\n    excerpt: source.excerpt || source.text || ''\n  }));\n}\n\n// Berechne Confidence Score\nif (response.confidence) {\n  processedResponse.confidence = response.confidence;\n} else if (response.score) {\n  processedResponse.confidence = response.score;\n} else if (processedResponse.sources.length > 0) {\n  // Durchschnittliche Relevanz als Confidence\n  const avgRelevance = processedResponse.sources.reduce((acc, s) => acc + s.relevance, 0) /
```

```
processedResponse.sources.length;\n  processedResponse.confidence =
avgRelevance;\n}\n\n// Formatiere die Antwort für bessere Lesbarkeit\nif
(processedResponse.entities.length > 0 ||
processedResponse.relationships.length > 0) {\n  let formattedAnswer =
processedResponse.answer + '\\n\\n';\n  \n  if
(processedResponse.entities.length > 0) {\n    formattedAnswer += '📊
**Identifizierte Entitäten:**\\n';\n
processedResponse.entities.forEach(entity => {\n      formattedAnswer += `•
${entity.name} (${entity.type})${entity.description ? ': ' +
entity.description : ''}\\n`;\n    });\n    formattedAnswer += '\\n';\n
}\n  \n  if (processedResponse.relationships.length > 0) {\n
formattedAnswer += '🔗 **Gefundene Beziehungen:**\\n';\n
processedResponse.relationships.forEach(rel => {\n      formattedAnswer +=
`• ${rel.source} → ${rel.type} → ${rel.target}\\n`;\n    });\n
formattedAnswer += '\\n';\n  }\n  \n  if (processedResponse.sources.length
> 0) {\n    formattedAnswer += '📚 **Quellen:**\\n';\n
processedResponse.sources.forEach((source, idx) => {\n      formattedAnswer
+= `${idx + 1}. ${source.document}${source.page ? ' (Seite ' + source.page
+ ')' : ''}\\n`;\n    });\n  }\n  \n  processedResponse.formattedAnswer =
formattedAnswer;\n}\n\n// Logging für Debugging\nconsole.log('Processed
Response:', JSON.stringify(processedResponse, null, 2));\n\nreturn
processedResponse;"
      },
      "id": "7dc91ff3-4e51-4b01-a156-4d126716d1c7",
      "name": "Process Graph Response",
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        1040,
        500
      ],
      "notesInFlow": false,
      "disabled": true
    },
    {
      "parameters": {
        "content": "## 📊 Optional: Response Processing\n\nDieser Code Node
kann aktiviert werden, um:\n- Graph-Daten zu strukturieren\n- Entitäten und
Beziehungen zu extrahieren\n- Antworten zu formatieren\n- Confidence Scores
zu berechnen\n\n**Aktivierung:**\n1. Rechtsklick auf den Node\n2.
\"Enable\" wählen\n3. Mit HTTP Request Tool verbinden",
        "height": 215.99999999999994,
        "width": 270.6666666666666,
        "color": 5
      },
      "id": "3a463bfd-80c6-42c9-ad96-3d8f804ba5bf",
      "name": "Sticky Note1",
      "type": "n8n-nodes-base.stickyNote",
      "typeVersion": 1,
      "position": [
        1180,
        80
      ]
    },
    {
      "parameters": {
        "httpMethod": "POST",
        "path": "graph-rag-webhook",
        "responseMode": "responseNode",
        "options": {
          "responseHeaders": {
            "entries": [
```

```
            {
              "name": "Access-Control-Allow-Origin",
              "value": "*"
            }
          ]
        }
      }
    },
    "id": "173448b8-e76a-47e9-95d7-8a14b8f6f961",
    "name": "Webhook (Alternative Trigger)",
    "type": "n8n-nodes-base.webhook",
    "typeVersion": 2,
    "position": [
      320,
      320
    ],
    "webhookId": "597de52e-d713-489b-9f0f-245237f26b9f",
    "disabled": true,
    "notes": "Alternative: Webhook für API Integration"
  },
  {
    "parameters": {
      "respondWith": "allIncomingItems",
      "options": {}
    },
    "id": "1bf886ec-128b-4862-93ee-f903a49cc0b6",
    "name": "Respond to Webhook",
    "type": "n8n-nodes-base.respondToWebhook",
    "typeVersion": 1.1,
    "position": [
      1260,
      620
    ],
    "disabled": true
  },
  {
    "parameters": {},
    "type": "@n8n/n8n-nodes-langchain.memoryBufferWindow",
    "typeVersion": 1.3,
    "position": [
      600,
      640
    ],
    "id": "10254d87-20b3-407d-b684-e48036f932db",
    "name": "Simple Memory"
  },
  {
    "parameters": {
      "options": {}
    },
    "type": "n8n-nodes-base.httpRequestTool",
    "typeVersion": 4.2,
    "position": [
      680,
      340
    ],
    "id": "172c0bed-937a-4ed5-b640-8adc6fab25bb",
    "name": "HTTP Request"
  }
],
"pinData": {},
"connections": {
  "When chat message received": {
```

```
    "main": [
      [
        {
          "node": "AI Agent",
          "type": "main",
          "index": 0
        }
      ]
    ]
  },
  "Webhook (Alternative Trigger)": {
    "main": [
      [
        {
          "node": "AI Agent",
          "type": "main",
          "index": 0
        }
      ]
    ]
  },
  "Process Graph Response": {
    "main": [
      [
        {
          "node": "Respond to Webhook",
          "type": "main",
          "index": 0
        }
      ]
    ]
  },
  "Simple Memory": {
    "ai_memory": [
      [
        {
          "node": "AI Agent",
          "type": "ai_memory",
          "index": 0
        }
      ]
    ]
  },
  "HTTP Request": {
    "ai_tool": [
      [
        {
          "node": "AI Agent",
          "type": "ai_tool",
          "index": 0
        }
      ]
    ]
  }
},
"active": false,
"settings": {
  "executionOrder": "v1"
},
"versionId": "WORKFLOW-VERSION-ID",
"meta": {
  "templateCredsSetupCompleted": true,
  "instanceId": "DEINE-N8N-INSTANCE-ID"
```

```
  },
  "id": "WORKFLOW-ID",
  "tags": []
}
```