Ontologien im Wandel

Impulsvortrag zum Ontologenkreis Darmstadt am 17. Juni 2024

Kai-Uwe Schäfer, Frankfurt am Main

Da es um den Titel dieses Impulsvortrags im Vorfeld einen konstruktiven Streit gab, der zeigt, dass jeder seine eigene Erwartungs-Ontologie heute Abend mitbringt, möchte ich Inhalt und Ziel zunächst deutlich machen.

- Manchmal wandeln sich Ontologien in Naturwissenschaft oder Gesellschaft inhaltlich (a) als Ausdruck von Anpassung an veränderte Umstände oder (b) als qualitative Weiterentwicklung (in Hegels Sinn). Darum soll es aber heute Abend nicht gehen.
- Es geht auch nicht um Ontologien des Wandels, gleichsam um Metamodelle für Veränderungsprozesse.
- Stattdessen soll es gehen um den sich wandelnden Einsatz von Methoden und Techniken zum Einsatz von Ontologien in der Informatik oder allgemeiner: der Wissensarbeit. Es müsste also eigentlich richtig heißen „der Einsatz von Ontologien im Wandel“.
- Gleichzeitig wollen wir uns aber auch austauschen über einen potenziellen Wandel des Ontologenkreises - wenn das nötig sein sollte.

Ontologien in der Krise

- Ich persönlich empfand die Arbeit mit Ontologien in der Informatik in den vergangenen Jahren bereits in einer Krise. Eine spezifische Ontologie zu erstellen - und sei es nur ein Thesaurus oder eine billige Taxonomie, um das Verständnis für eine Problemdomäne zu verbessern - findet keine Freunde. Ontologien sind etwas für Menschen, die es lieben, Ordnung zu schaffen, zu sammeln, akademisch zu arbeiten, „das Richtige zu tun“. **Ontologien sind nichts für enge Budgets und Zeitpläne**.
- Ich erinnere mich noch an das Resümee unseres Kollegen Christian Fillies, der meinte, mit Ontologien lasse sich einfach kein Geld verdienen. Oder auch an Hermann Bense, der mit großer Ausdauer viele tausend Tripel auf Basis eines gut durchdachten Metamodells erarbeitet, um Alltagswissen greifbar zu machen. Darin steckt ausgesprochen viel Arbeit und Anstrengung.
- Das sicherlich größte Ontologie-Projekt war das Projekt “Cyc” von Douglas Lenat. Gestartet 1984, versuchte man, mit einem Team von Ontologen, Alltagswissen maschinenlesbar zu sammeln und aufzubereiten. An dem Projekt wird nach 40 Jahren noch gearbeitet, aber es gilt praktisch als gescheitert. Die Menge und Komplexität des menschlichen Wissens mit seinen Unschärfen, Nuancen und Kontexten war manuell nicht beherrschbar und nicht skalierbar. Details zur Fallstudie siehe Anhang 1.
- Selbst in strategischen Disziplinen wie der Unternehmensarchitektur haben es Ontologien schwer: es will keiner mitmachen oder sich auch nur in das enge Korsett einer fremden Bedeutungshoheit hineinzwängen, nämlich dem viel kritisierten Elfenbeinturm. **Ontologien sind starr und nicht anpassungsfähig**. Diese Unternehmensmodelle leiden allerdings auch sehr unter Trends: erst Prozessorientierung, dann Business Capabilities, dann Daten getriebenes Unternehmen - und jedesmal eine Umorganisation anhand der neuen Ontologie.

- Antithese: **Ontologien sind langfristig stabil und verlässlich. Ontologien schaffen Standards und Grundlagen**. Leider ist das IT-Projektgeschäft nicht an langfristigen Werten interessiert, sondern an kurzfristigen Lösungen, die sich erwartungsgemäß relativ schnell wieder wandeln.
- Wo sind also die Anwendungsfälle, die stabile und verlässliche Grundlagen benötigen? Und **wo sind die Ontologien, die sich nicht ständig wandeln müssen?**

Ontologien in Konkurrenz

Ontologien in der Informatik sollen helfen, für die Maschine ein Verständnis von der Welt zu schaffen. Als LLMs aufkamen, schien die Arbeit an Ontologien plötzlich rechts überholt zu werden. Hatten wir die Semantik in einer Ontologie mit Intelligenz verwechselt und daher zu hohe Erwartungen an ihren Mehrwert und ihre Einsatzgebiete?

Vergleich subsymbolische versus symbolische Verfahren

Jetzt kommen plötzlich subsymbolische Verfahren wie Large Language Models (LLM) mit einem erstaunlich guten Textverständnis um die Ecke - auch wenn es sich nicht im engeren Sinn um Verstehen handelt. Wofür also noch Ontologien, wenn das doch alles mit KI jetzt so viel schneller geht?

Symbolische Verfahren können mit subsymbolischen kombiniert werden, um deren Ergebnisse zu strukturieren und zu erklären bzw. überhaupt zu strukturierten Ergebnissen zu führen.

Kategorie	Subsymbolische Verfahren	Symbolische Verfahren
Datenmenge	groß	klein
Verfahren	Maschinelles Lernen, Statistik	formale Logik
Ergebnistypen	Muster, Vorhersagen	Ontologien, Knowledge Graphs
Flexibilität	anpassungsfähig	
Unschärfen möglich	starr	
keine Mehrdeutigkeiten		
Zuverlässigkeit	schwer interpretierbar	
unscharf	gut nachvollziehbar	
verlässlich		
Konkretheit	nur statistische Modelle	Abstraktion möglich
Beziehungen	keine expliziten	komplexe möglich
Anwendbarkeit	auf Trainingsdaten begrenzt	flexibel in neuen Situationen

Ontologien in Zusammenarbeit mit LLM

Retrieval Augmented Generation (RAG)

"Retrieval Augmented Generation" (RAG) ist eine Technik im Bereich der Künstlichen Intelligenz (KI) und des Natural Language Processing (NLP), die Textgenerierungsmodelle mit externen Wissensdatenbanken kombiniert. Die Methode besteht aus zwei Hauptkomponenten:

1. **Retrieval (Abruf)**: Ein System durchsucht eine externe Datenbank oder ein Wissensarchiv nach relevanten Informationen, die für eine bestimmte Anfrage nützlich sein könnten. Diese Datenbank kann eine Sammlung von Dokumenten, Webseiten oder eine andere Art von Wissensquelle sein.
2. **Generation (Generierung)**: Ein generatives Sprachmodell, wie GPT (Generative Pre-trained Transformer), verwendet die abgerufenen Informationen, um eine kohärente und kontextuell relevante Antwort zu erstellen.

Siehe Anhang 2 für weitere Details.

Retrieval Augmented Generation (RAG) auf Ontologien

RAG auf umfangreichen Ontologien stellt eine substanzielle Optimierung dar:

- **Präzisere Antworten**: Durch die strukturierte und explizite Repräsentation von Wissen können Antworten genauer und relevanter sein.
- **Erweiterte Kontextualisierung**: Ontologien ermöglichen eine tiefere Kontextualisierung, da sie die Beziehungen zwischen Konzepten klar definieren.
- **Effiziente Informationssuche**: Ontologien können die Effizienz der Informationssuche erhöhen, indem sie relevante Konzepte und Beziehungen schnell identifizieren.

Siehe Anhang 3 für weitere Details.

Der Autor von Understanding Retrieval-Augmented Generation (RAG) empowering LLMs nennt das Beispiel einer Dokumenten-Datenbank, deren Inhalte in Vektoren übersetzt (Embeddings) und in eine Vektordatenbank eingetragen werden. Ein Wissensgraph kann nach dem gleichen Prinzip in Vektoren umgewandelt werden. Neo4J hat das mittlerweile eingebaut. Dafür würde im Schaubild des Artikels „Documents“ durch „Knowledge Graph“ ersetzt.

Ein offensichtlich deutlich besserer Schritt - wenn auch komplizierter - ist die Auswertung eines Knowledge Graph mit Graph Queries, um einen Query Context für LLMs zu erstellen. Der Vorgang wird manchmal GraphRAG genannt. Einige Artikel zu GraphRAG wie folgt. Besonders die ersten beiden Artikel zeigen verständliche Beispiele:

- Using a Knowledge Graph to Implement a RAG Application
- Knowledge Graph vs. Vector RAG: Benchmarking, Optimization Levers, and a Financial Analysis Example
- From Conventional RAG to Graph RAG
- Enhancing the Accuracy of RAG Applications With Knowledge Graphs
- Knowledge Graphs for Retrieval-Augmented Generation (RAG)
- Graph RAG vs Vector RAG: A Comprehensive Tutorial with Code Examples

Wichtige Erkenntnis: RAG auf Basis einer Vektor-DB ist gut geeignet, um Ähnlichkeiten oder implizite Beziehungen in einer Query zu berücksichtigen. Aber diese Ergebnisse sind weder vollständig (im Sinne der Datenbasis), noch exakt, noch in jedem Fall sinnvoll, da Cosinus-Ähnlichkeiten nur Näherungen beisteuern, die je nach Kontext nicht gut anwendbar sind.

Queries in einem Knowledge Graph sind hingegen vollständiger, da sie den kompletten Kontext in der Datenbasis berücksichtigen können und sind außerdem exakter, da sie nicht auf Näherungen, sondern auf expliziten Relationen basieren.

Wissensgraph um Allgemeinwissen ergänzen

In diesem Artikel wird umgekehrt ein Knowledge Graph ergänzt um Alltagswissen, welches per LLM aus Texten herausgelesen wurde: Building commonsense knowledge graphs to aid product recommendation. Hier wurden Tripels für einen Graphen erzeugt, indem Aussagen aus Produktbewertungen per NLP ausgewertet und in den Wissensgraph integriert wurden. Beispiel: “Habe mir die Schuhe gekauft, weil ich mich als Schwangere mit rutschfesten Gummisohlen sicherer fühle.” → Tripel(pregnant, require, slip-resistant)

Neue Werkzeuge zur Erstellung und Pflege von Ontologien

Large Language Models (LLM) können die Erstellung und Pflege von Ontologien unterstützen. Neue Werkzeuge und Plattformen zur Erstellung, Verwaltung und Nutzung von Ontologien werden die Barrieren für den Einsatz symbolischer Verfahren senken und deren Verbreitung fördern. Dies wird es mehr Menschen und Organisationen ermöglichen, von den Vorteilen der Ontologien zu profitieren.

- Unterstützung bei der Erstellung von Ontologien
- Methoden zur automatischen Erweiterung und Verfeinerung von Ontologien
- Methoden zur kontinuierlichen Aktualisierung und Anpassung von Ontologien an sich ändernde Wissensdomänen

Austausch im Auditorium

Impulse: Philosophie, Linguistik, Interaktion Mensch-Maschine, Rechtssystem, Produkte am Markt, Psychologie, Mathematik, Öffentliche Verwaltung, Gesundheitssystem, Informatik, …

Anhang 1: Woran war das Projekt “Cyc” gescheitert?

Der folgende Text wurde mittels ChatGPT 4o weitestgehend generiert.

“Intelligence is 10 million rules” (Douglas Lenat)

Das Cyc-Projekt, initiiert von Doug Lenat in den 1980er Jahren, verfolgte das Ziel, eine umfassende Wissensbasis zu schaffen, die Computer befähigen sollte, menschliche Sprache und Konzepte zu verstehen und darauf zu reagieren. Trotz der ambitionierten Vision scheiterte das Projekt aus mehreren Gründen:

1. Umfang und Komplexität: Das Ziel, eine umfassende Wissensbasis zu erstellen, erwies sich als enorm herausfordernd. Die Menge an Wissen, die codiert werden musste, war überwältigend, und es war schwierig, alles abzudecken.
2. Manuelle Wissenseingabe: Das Wissen wurde hauptsächlich manuell durch Experten eingegeben, was zeitaufwändig und kostspielig war. Dieser Prozess war nicht nur langsam, sondern auch fehleranfällig und inkonsistent.
3. Unzureichende Skalierbarkeit: Das Cyc-System hatte Schwierigkeiten, sich auf neue Domänen und Wissen zu skalieren. Es war schwierig, das System an die sich schnell verändernde Wissenslandschaft anzupassen.
4. Fehlende Integration mit maschinellem Lernen: Während das Cyc-Projekt hauptsächlich auf regelbasiertes Wissen setzte, gewannen maschinelle Lernansätze an Bedeutung. Cyc konnte nicht effektiv mit diesen neueren Technologien integriert werden, die sich als leistungsfähiger in der Handhabung großer Datenmengen und unscharfer Informationen erwiesen.
5. Komplexität der natürlichen Sprache: Die natürliche Sprache ist extrem komplex und mehrdeutig. Cyc hatte Schwierigkeiten, mit den vielfältigen Nuancen, Kontexten und Bedeutungen menschlicher Sprache umzugehen.
6. Wirtschaftliche und praktische Herausforderungen: Es war schwierig, kurzfristig wirtschaftliche Anwendungen und Vorteile aus dem Projekt zu ziehen. Investoren und Unternehmen hatten Schwierigkeiten, den praktischen Nutzen und die Rentabilität des Projekts zu sehen, was die Finanzierung und langfristige Unterstützung erschwerte.

Insgesamt war das Cyc-Projekt ein visionäres Unterfangen, das viele Grundlagen für spätere Entwicklungen im Bereich der Künstlichen Intelligenz legte. Die Herausforderungen und Grenzen, auf die Cyc stieß, zeigten jedoch auch die Schwierigkeiten auf, eine umfassende und allumfassende Wissensbasis zu schaffen, die mit der Komplexität menschlichen Wissens und der natürlichen Sprache umgehen kann.

Laufzeit des Projekts

Das Cyc-Projekt begann 1984 und läuft im Grunde bis heute weiter. Es wurde von Doug Lenat und seiner Firma Cycorp ins Leben gerufen. Obwohl das Projekt in den ersten Jahrzehnten mit erheblichen Herausforderungen und Kritik konfrontiert war, hat es nie offiziell aufgehört zu existieren und wurde kontinuierlich weiterentwickelt.

Hier ist ein grober Überblick über die Laufzeit des Projekts:

1. 1984: Beginn des Cyc-Projekts.
2. 1994: Zehn Jahre nach Beginn des Projekts wurde Cycorp gegründet, um die Arbeit an Cyc fortzusetzen und kommerzielle Anwendungen zu entwickeln.
3. 2001: OpenCyc wurde eingeführt, eine Open-Source-Version der Cyc-Wissensbasis, um eine breitere Nutzung und Zusammenarbeit zu fördern.

4. 2012: Eine weitere Open-Source-Version, genannt Freebase, basierend auf Cyc-Daten, wurde von Google übernommen und in das Knowledge Graph-Projekt integriert.
5. Heute: Cycorp arbeitet weiterhin an der Weiterentwicklung von Cyc und seiner Anwendungen.

Das Cyc-Projekt hat somit eine Laufzeit von über 40 Jahren und wird immer noch aktiv weiterentwickelt, auch wenn es im Laufe der Zeit seine Schwerpunkte und Ansätze teilweise angepasst hat.

Anhang 2: Retrieval Augmented Generation (RAG)

"Retrieval Augmented Generation" (RAG) ist eine Technik im Bereich der Künstlichen Intelligenz (KI) und des Natural Language Processing (NLP), die Textgenerierungsmodelle mit externen Wissensdatenbanken kombiniert. Die Methode besteht aus zwei Hauptkomponenten:

1. **Retrieval (Abruf)**: Ein System durchsucht eine externe Datenbank oder ein Wissensarchiv nach relevanten Informationen, die für eine bestimmte Anfrage nützlich sein könnten. Diese Datenbank kann eine Sammlung von Dokumenten, Webseiten oder eine andere Art von Wissensquelle sein.
2. **Generation (Generierung)**: Ein generatives Sprachmodell, wie GPT (Generative Pre-trained Transformer), verwendet die abgerufenen Informationen, um eine kohärente und kontextuell relevante Antwort zu erstellen.

Der Ablauf sieht in der Regel wie folgt aus:

1. **Benutzeranfrage**: Eine Anfrage oder Frage wird an das System gestellt.
2. **Information Retrieval**: Das System durchsucht eine große Menge von Dokumenten oder eine spezialisierte Wissensdatenbank, um die relevantesten Informationen zur Anfrage zu finden.
    - **Initiales Retrieval**: Das System verwendet ein Retrieval-Modell, um relevante Dokumente oder Textpassagen aus einer großen Sammlung von Texten (z.B. Wikipedia, wissenschaftliche Artikel, Datenbanken) zu suchen. Dies geschieht oft mithilfe von Dense Passage Retriever (DPR) oder ähnlichen Technologien, die semantische Ähnlichkeiten erkennen. Siehe auch Abschnitt ‘Arbeitsschritt “Embeddings erzeugen”’.
    - **Auswahl der relevantesten Passagen**: Die abgerufenen Dokumente oder Textpassagen werden anhand ihrer Relevanz bewertet. Dies kann durch Ranking-Algorithmen geschehen, die beispielsweise die Ähnlichkeit zur Benutzeranfrage messen.
    - **Kombinieren der abgerufenen Passagen**: Die ausgewählten relevanten Textpassagen werden in einer strukturierten Weise kombiniert. Hierbei können verschiedene Strategien angewendet werden: Verkettung, Zusammenfassung, Gewichtung und Auswahl.
3. **Information Integration**: Die abgerufenen Informationen werden in das Sprachmodell eingespeist.
    - Die zusammengeführten Informationen werden dem generativen Sprachmodell als Eingabe übergeben. Dies geschieht oft in Form eines erweiterten Kontexts, in den die abgerufenen Texte eingefügt werden. Das Sprachmodell erhält also nicht nur die ursprüngliche Anfrage des Benutzers, sondern auch die relevanten zusätzlichen Informationen aus der Datenbank.
4. **Antwortgenerierung**: Das Sprachmodell generiert eine Antwort, die auf den abgerufenen Informationen basiert und diese integriert.

Diese Technik verbessert die Genauigkeit und Relevanz der generierten Antworten, insbesondere in Bereichen, wo detaillierte und spezifische Informationen benötigt werden. RAG kombiniert also die Stärken von Suchalgorithmen (die Fähigkeit, relevante Informationen aus großen Datenmengen abzurufen) mit den Stärken von Sprachmodellen (die Fähigkeit, menschenähnlichen Text zu generieren).

Ein Beispiel für die Anwendung von RAG ist ein Frage-Antwort-System, das wissenschaftliche Artikel durchsucht, um präzise Antworten auf komplexe wissenschaftliche Fragen zu geben. Ein weiteres Beispiel sind Chatbots, die auf umfangreiche Wissensdatenbanken zugreifen, um den Nutzern fundierte und genaue Auskünfte zu geben.

Arbeitsschritt “Embeddings erzeugen”

Neuronale Netzwerke, die dichte Vektor-Repräsentationen (Embeddings) von Dokumenten erzeugen, funktionieren durch die Transformation von Text in numerische Vektoren in einem kontinuierlichen Vektorraum. Diese Vektoren fassen die semantischen Informationen des Textes zusammen und ermöglichen es, semantische Ähnlichkeiten zwischen Texten effizient zu berechnen. Hier ist eine detaillierte Erklärung des Prozesses:

1. Eingabevorverarbeitung

Die Eingabe, d.h. der Text des Dokuments oder der Anfrage, wird zunächst vorverarbeitet:

- **Tokenisierung**: Zerlegung des Textes in kleinere Einheiten (Tokens), wie Wörter oder Subwörter.
- **Normalisierung**: Umwandlung in Kleinbuchstaben, Entfernung von Satzzeichen, etc.
- **Stoppwort-Entfernung** (optional): Entfernen häufig vorkommender, wenig informative Wörter (wie "und", "der", "ist").

2. Einbettung von Wörtern (Word Embeddings)

Jedes Token wird in einen Vektor umgewandelt. Dies geschieht mit vortrainierten Einbettungstechniken wie Word2Vec, GloVe oder neueren Methoden wie Transformer-basierten Modellen (z.B. BERT, GPT):

- **Word2Vec/GloVe**: Wörter werden durch dichte Vektoren dargestellt, die in einem Trainingsprozess gelernt wurden, indem sie die kontextuelle Umgebung der Wörter berücksichtigt haben.
- **Transformer-Modelle**: Modelle wie BERT (Bidirectional Encoder Representations from Transformers) oder GPT (Generative Pre-trained Transformer) erzeugen kontextabhängige Wortvektoren, die den gesamten Kontext eines Satzes berücksichtigen.

3. Architektur des Neuronalen Netzwerks

Die spezifische Architektur des neuronalen Netzwerks hängt vom Modell ab, das zur Erzeugung der dichten Vektor-Repräsentationen verwendet wird. Hier sind einige der gebräuchlichsten Architekturen:

- A. **Recurrent Neural Networks (RNNs)**
  - **LSTM (Long Short-Term Memory)** und **GRU (Gated Recurrent Units)** sind Varianten von RNNs, die genutzt werden, um Sequenzen von Wörtern zu verarbeiten und kontextuelle Informationen über längere Sequenzen zu behalten.
- B. **Convolutional Neural Networks (CNNs)**
  - CNNs können auf Text angewendet werden, um lokale Features in Textfenstern zu erkennen und zu kombinieren, was besonders nützlich für die Erfassung von lokalen Kontexten ist.
- C. **Transformer-basierte Modelle**
  - **Transformer-Modelle** wie BERT und GPT verwenden Selbstaufmerksamkeit (Self-Attention) Mechanismen, um kontextuelle Informationen aus dem gesamten Text gleichzeitig zu berücksichtigen, was sie besonders effektiv macht für NLP-Aufgaben.

4. Erzeugung der Dokument- oder Passage-Embedding

Nachdem Wort-Embeddings erzeugt wurden, müssen sie zu einer einzigen, festen Länge Vektor-Repräsentation des gesamten Dokuments oder der Passage zusammengefasst werden:

- **Pooling-Methoden**:

- **Max-Pooling**: Der maximale Wert für jedes Feature über alle Token hinweg wird gewählt.
- **Mean-Pooling**: Der Durchschnittswert für jedes Feature über alle Token hinweg wird berechnet.
- **CLS-Token**: Bei Modellen wie BERT wird das spezielle [CLS]-Token verwendet, dessen Einbettung als Repräsentation des gesamten Eingabetextes dient.

5. Training des Modells

Das Modell wird mit einem speziellen Trainingsprozess optimiert, um sicherzustellen, dass die erzeugten Vektoren semantische Ähnlichkeiten gut erfassen. Es gibt mehrere gängige Trainingsmethoden:

- **Contrastive Learning**: Das Modell lernt, ähnliche Paare von Texten (z.B. Frage und passende Antwort) näher zusammenzubringen und unähnliche Paare weiter auseinander.
- **Triplet Loss**: Das Modell wird darauf trainiert, die Distanz zwischen einem Ankertext und einem positiven Beispiel zu minimieren, während es die Distanz zu einem negativen Beispiel maximiert.
- **Supervised Learning**: Bei Aufgaben wie Frage-Antwort-Paaren oder Information Retrieval wird das Modell direkt darauf trainiert, passende Paare korrekt zu identifizieren.

6. Berechnung von Ähnlichkeiten

Sobald die Vektor-Repräsentationen für die Dokumente erzeugt wurden, können sie verwendet werden, um Ähnlichkeiten zu berechnen. Die gängigste Methode ist die Berechnung der Kosinusähnlichkeit zwischen den Vektoren.

Beispiel: BERT zur Erzeugung von Embeddings

1. **Eingabe**: "Wie funktioniert Photosynthese?"
2. **Tokenisierung**: ["[CLS]", "Wie", "funktioniert", "Photosynthese", "?", "[SEP]"]
3. **Word Embeddings**: BERT erzeugt kontextuelle Embeddings für jedes Token.
4. **Pooling**: Die Einbettung des [CLS]-Tokens wird als Repräsentation des gesamten Satzes verwendet.
5. **Ergebnis**: Ein fester Längenvektor, der die Bedeutung des gesamten Textes zusammenfasst.

Fazit

Neuronale Netzwerke, die dichte Vektor-Repräsentationen erzeugen, transformieren Textdaten in numerische Vektoren, die semantische Informationen enthalten. Dies wird durch komplexe Architekturen wie Transformer-Modelle und spezialisierte Trainingsverfahren erreicht, die sicherstellen, dass ähnliche Texte im Vektorraum nahe beieinander liegen, was eine effiziente Berechnung von Textähnlichkeiten ermöglicht.

Anhang 3: RAG auf Ontologien

Retrieval Augmented Generation (RAG) kann auch auf Ontologien basieren. Ontologien bieten eine strukturierte und explizite Darstellung von Wissen in einem bestimmten Domänenbereich und können dazu beitragen, den Retrieval- und Generierungsprozess zu verbessern. Hier ist, wie RAG auf Ontologien angewendet werden kann:

Was ist eine Ontologie?

Eine Ontologie ist eine formale Repräsentation von Wissen als eine Sammlung von Konzepten innerhalb einer Domäne und den Beziehungen zwischen diesen Konzepten. Sie umfasst:

- **Klassen (Konzepte)**: Grundlegende Einheiten wie "Mensch", "Pflanze", "Tier".
- **Instanzen**: Konkrete Beispiele dieser Klassen wie "John", "Eiche".
- **Eigenschaften**: Attribute oder Merkmale der Klassen und Instanzen.
- **Beziehungen**: Verbindungen zwischen den Konzepten, wie "ein Mensch hat ein Herz" oder "eine Pflanze führt Photosynthese durch".

Integration von Ontologien in RAG

Die Integration von Ontologien in den RAG-Ansatz kann die Qualität und Relevanz der abgerufenen Informationen und der generierten Antworten erheblich verbessern. Hier ist ein Überblick über den Prozess:

1. Benutzeranfrage

Der Benutzer stellt eine Frage oder Anfrage, die an das System weitergeleitet wird.

2. Semantische Annotation

Die Anfrage wird semantisch annotiert, um die relevanten Konzepte und Beziehungen in der Ontologie zu identifizieren. Dies könnte durch Techniken wie Named Entity Recognition (NER) und Part-of-Speech Tagging (POS) erfolgen.

3. Initiales Retrieval basierend auf Ontologie

Anstatt nur auf Textdatenbanken zuzugreifen, verwendet das System die Ontologie, um relevante Konzepte und Beziehungen abzurufen. Dies kann durch folgende Schritte erfolgen:

- **Ontologie-basierte Abfrageerweiterung**: Die ursprüngliche Anfrage wird erweitert, indem synonyme und verwandte Begriffe aus der Ontologie hinzugefügt werden.
- **Ontologie-basierte Suche**: Die Ontologie wird durchsucht, um relevante Klassen, Instanzen und Beziehungen zu identifizieren, die zur Beantwortung der Anfrage beitragen können.

4. Extraktion relevanter Informationen

Die Informationen, die durch die Ontologie identifiziert wurden, werden aus den entsprechenden Dokumenten oder Wissensquellen extrahiert. Die Ontologie hilft dabei, die relevantesten Passagen und Datenpunkte zu finden.

5. Erweiterter Kontext

Der erweiterte Kontext wird nicht nur durch textbasierte Informationen, sondern auch durch die strukturierten Daten der Ontologie ergänzt. Dies könnte beinhalten:

- **Direkte Antworten**: Informationen direkt aus der Ontologie, wie definierte Eigenschaften oder Beziehungen.

- **Kontextuelle Informationen**: Zusätzliche Informationen aus Textdokumenten, die durch die Ontologie als relevant identifiziert wurden.

6. Generierung der Antwort

Das generative Modell erhält den erweiterten Kontext, der sowohl textbasierte als auch ontologiebasierte Informationen enthält. Das Modell integriert diese Informationen, um eine kohärente und präzise Antwort zu generieren.

Beispielprozess mit Ontologie

1. **Benutzeranfrage**: "Wie funktioniert Photosynthese bei Pflanzen?"
2. **Semantische Annotation**: Identifikation der Konzepte "Photosynthese" und "Pflanzen" in der Ontologie.
3. **Ontologie-basierte Abfrageerweiterung**: Erweiterung der Anfrage um synonyme und verwandte Begriffe wie "Lichtreaktion", "Calvin-Zyklus", "Chloroplast".
4. **Ontologie-basierte Suche**: Durchsuchen der Ontologie nach relevanten Konzepten und Beziehungen, wie "Photosynthese → Lichtreaktion", "Pflanze → Chloroplast".
5. **Extraktion relevanter Informationen**: Abrufen von Textpassagen und definierten Informationen, wie die Schritte der Lichtreaktion und des Calvin-Zyklus aus einer Wissensdatenbank.
6. **Erweiterter Kontext**: Kombination der textbasierten Informationen mit den strukturierten Daten aus der Ontologie.
7. **Generierung der Antwort**: Das Sprachmodell erzeugt eine Antwort, die sowohl die detaillierten Prozesse der Photosynthese als auch die kontextuellen Beziehungen berücksichtigt.

Vorteile der Verwendung von Ontologien in RAG

- **Präzisere Antworten**: Durch die strukturierte und explizite Repräsentation von Wissen können Antworten genauer und relevanter sein.
- **Erweiterte Kontextualisierung**: Ontologien ermöglichen eine tiefere Kontextualisierung, da sie die Beziehungen zwischen Konzepten klar definieren.
- **Effiziente Informationssuche**: Ontologien können die Effizienz der Informationssuche erhöhen, indem sie relevante Konzepte und Beziehungen schnell identifizieren.

Fazit

Die Integration von Ontologien in Retrieval Augmented Generation kann die Effizienz und Genauigkeit von Antwortgenerierungssystemen erheblich verbessern. Ontologien bieten eine strukturierte Wissensbasis, die es ermöglicht, präzisere und kontextuell reichere Antworten zu generieren. Dies ist besonders nützlich in spezialisierten Domänen, wo detailliertes und strukturiertes Wissen erforderlich ist.