Retrieval-Augmented Generation - Enhancement 1

SOTA2 - Group C

Anton Kesy, Étienne Muser, Katharina Schindler, Lukas Fehrenbacher, Nico Ruschmann

Offenburg University of Applied Sciences

WS 2024/2025

Retrieval-Augmented Generation - Eingabeverbesserung

- generierung von Inhalten mithilfe von Informationen aus einem externen Datenspeicher verbessert
- verschiedene Methoden zur Verbesserung der Eingabe und des Retrievers in RAG-Systemen.

Abfrage-Transformation

- modifiziert die Eingangsabfrage, um die Qualität der abgerufenen Inhalte zu verbessern.

Daten-Augmentierung

- Daten-Augmentierung verbessert die Daten vor dem Abruf:
  - durch Entfernen irrelevanter oder mehrdeutiger Informationen
  - Aktualisieren veralteter Dokumente
  - Synthetisieren neuer Daten.

Eingabeverbesserung - Abfrage-Transformation:

- **Query2doc** und **HyDE**
  - verwenden die ursprüngliche Abfrage, um ein Pseudo-Dokument zu erstellen, das dann als Abfrage für den Retriever dient.
- **TOC**
  - zerlegt mehrdeutige Abfragen mithilfe von abgerufenen Inhalten in mehrere eindeutige Unterabfragen.
- **RQ-RAG**
  - zerlegt komplexe oder mehrdeutige Abfragen in klare Unterabfragen für eine fein abgestufte Suche.
- **Tayal**
  - verfeinern die ursprüngliche Abfrage mit dynamischen Beispielen und Kontextinformationen.

Eingabeverbesserung - Daten-Augmentierung:

- **Make-An-Audio**
  - generiert Bildunterschriften für sprachfreie Audiodaten, um Datenknappheit zu beheben, und fügt zufällige Konzept-Audiodaten hinzu, um die ursprünglichen Audiodaten zu verbessern.
- **LESS**
  - optimiert die Datenauswahl durch Analyse von Gradienteninformationen.
- **ReACC**
  - verwendet Daten-Augmentierung (einschliesslich Umbenennung und Einfügen von totem Code) für das Vortraining des Code-Retrieval-Modells.
- **Telco-RAG**
  - verbessert die Abrufgenauigkeit durch Anwendung eines "Vokabulars für 3GPP-Spezifikationen" und dessen Abgleich mit Benutzerabfragen über ein Router-Modul.

Retreiver Enhancements (1/2)

Recursive Retreival

- Mehrere Retreivals
- z.B. Chain of Thought-mäßig

Chunk Optimization

- Optimieren der große der retreiveten chunks
- RAPTOR - Später

Retreiver Finetuning

- Retreiver Modell finetunen

Retreiver Enhancements (2/2)

Hybrid Retreival

- Mehrere Retreiver verwenden
- Blended Rag - Später

Re-Ranking

- Retreiveter Content neu anordnen

Retreival Transformation

- Retreiveten Content umformulieren

Blended RAG

Ziel: Verbesserung der Genauigkeit von RAG Systemen durch die Kombination von semantischen Suchtechniken und hybriden Abfragestrategien.

- Suchstrategien
  - Keyword basierte Ähnlichkeitssuche
  - dichte vektorbasierte Suche
  - semantische Suche mit Sparse Encodern
- Hybride Abfragen kombinieren Suchtechniken, um Stärken zu nutzen und Retrieval-Genauigkeit zu verbessern
- Hybride Abfragen
  - Cross Fields
  - Most Fields
  - Phrase Prefix
  - ...
- Beste Abfragen werden verwendet, um generative Frage-Antwort-Systeme zu speisen und die genauesten Antworten zu generieren

Vor- und Nachteile von hybridem Retrieval

Vorteile

- Verbesserte Retrieval-Genauigkeit
  - Hybride Abfragen können die verfügbaren Metadaten effektiver nutzen, um die relevantesten Ergebnisse zu finden
- Nutzung semantischer Beziehungen
  - Semantische Suchmethoden liefern oft bessere Ergebnisse als herkömmliche Suchmethoden
- Effizienzsteigerung
  - Hybride Abfragen kombinieren die Vorteile verschiedener Index-Typen
  - Optimiert Indexierungs- und auch die Abfragegeschwindigkeit

Nachteile

- Abhängigkeit von Metadaten
  - Effektivität ist stark von der Verfügbarkeit und Qualität von Metadaten abhängig.
  - In Datensätzen ohne Metadaten bieten hybride Abfragen keinen signifikanten Vorteil gegenüber einfachen Abfragen
- Komplexität und Rechenaufwand
  - Die Implementierung kann komplex und rechenintensiv sein, insbesondere bei großen Datensätzen

Ausgangsproblem

Problem: Die meisten abrufgestützen Sprachmodelle können nur kurze, zusammenhängende Textfragmente abrufen. Das schränkt ihre Fähigkeit ein, komplexe Themen zu verstehen und zu nutzen, die aus mehreren Teilen eines Textes abgeleitet werden müssen.

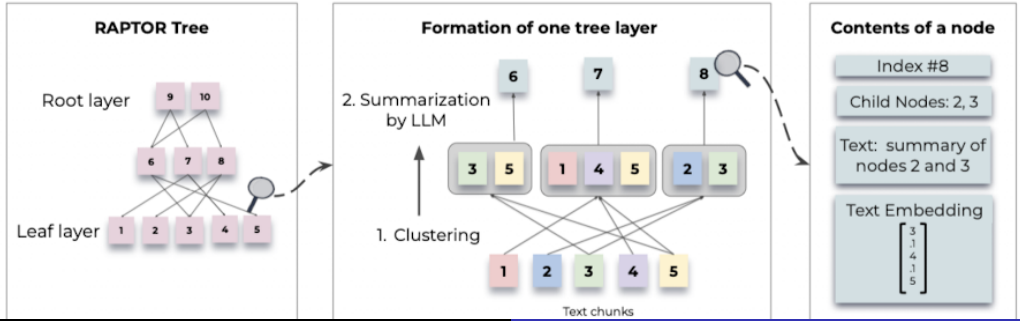
Beispiel: Um die Frage “Wie erreichte Aschenputtel ihr Happy End?” anhand des Märchens von Aschenputtel zu beantworten, würden die Top-k abgerufenen kurzen, zusammenhängenden Texte nicht genügend Kontext liefern.

Dieses Problem ist besonders relevant für thematische Fragen, die die Integration von Wissen aus mehreren Teilen eines Textes erfordern

-> RAPTOR löst dieses Problem, indem es eine Baumstruktur verwendet, die sowohl übergeordnete als auch untergeordnete Details eines Textes erfasst

Tree construction process

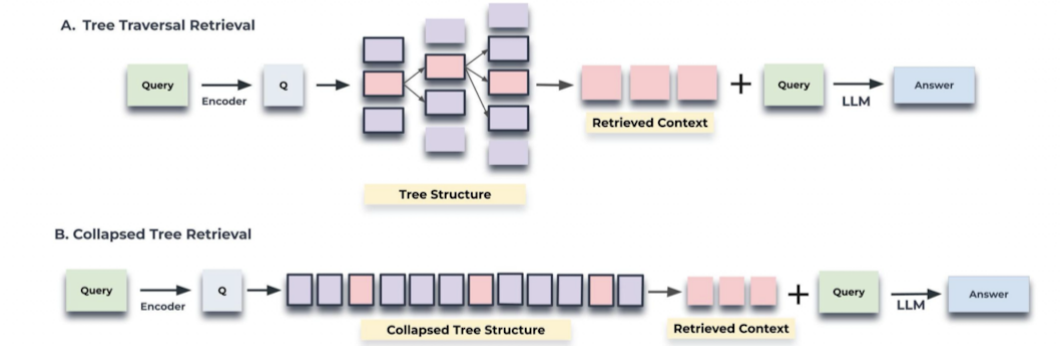
1. Segmentierung: verarbeitenden Texte in kurze, zusammenhängende Textsegmente mit einer Länge von 100 Token aufteilen
2. Einbettung: mithilfe von SBERT in Vektoren umgewandelt (bilden die Blattknoten des Baums)
3. Clustering: mit Soft Clustering, bei dem Knoten zu mehreren Clustern gehören können -> Besonders wichtig, da Textsegmente oft Informationen enthalten, die für verschiedene Themen relevant sind.



Tree traversal and collapsed tree retriveal mechanism

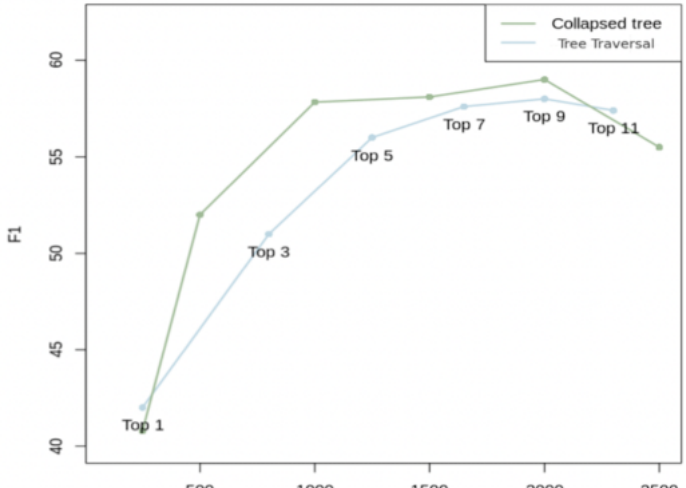
Tree traversal: Baumstruktur Schicht für Schicht durchlaufen und auf jeder Ebene die relevantesten Knoten ausgewählen

Tree retriveal: Gesamte Baumstruktur auf einmal betrachten und die relevantesten Knoten auswählen (alle Knoten auf eine Schicht zusammenfassen)



Vergleich Tree traversal and collapsed tree retriveal mechanism

In der Praxis hat sich der collapsed tree retriveal Mechanismus als am effektivsten erwiesen



Herausforderungen beim Junking im RAPTOR-Ansatz

1. **Irrelevanz und Datenrauschen**
   - Aufnahme redundanter oder irrelevanter Informationen in die Baumstruktur.
   - Erschwert die Fokussierung auf wesentliche Inhalte.
2. **Qualitätsverlust durch mehrstufige Verarbeitung**
   - Präzisionsverlust bei jedem Clustering- und Summarisierungsschritt.
   - Risiko von verwässerten Details und Zusammenhängen.
3. **Schwierigkeiten bei der Kontextwahrung**
   - Semantische Nähe != inhaltlicher Zusammenhang.
   - Entfernte, aber relevante Passagen können übersehen werden.

Weitere Herausforderungen und Lösungsansätze

4. **Kognitive Belastung für das Modell**
   - Verarbeitung von Junk-Daten erhöht Rechenaufwand und Antwortzeit.
5. **Kosten und Skalierung**
   - Höherer Rechenaufwand durch irrelevante Daten in größeren Datensätzen.
6. **Abhängigkeit von Summarisierung**
   - Suboptimale Zusammenfassungen können wichtige Details verlieren.

Lösungsansätze:

- Fortschrittliche Filtermethoden und Kontextanalyse.
- Optimierung von Clustering und Summarisierung.
- Kontinuierliche Qualitätsprüfung der Baumstruktur.