RAG for Source Text Summarization

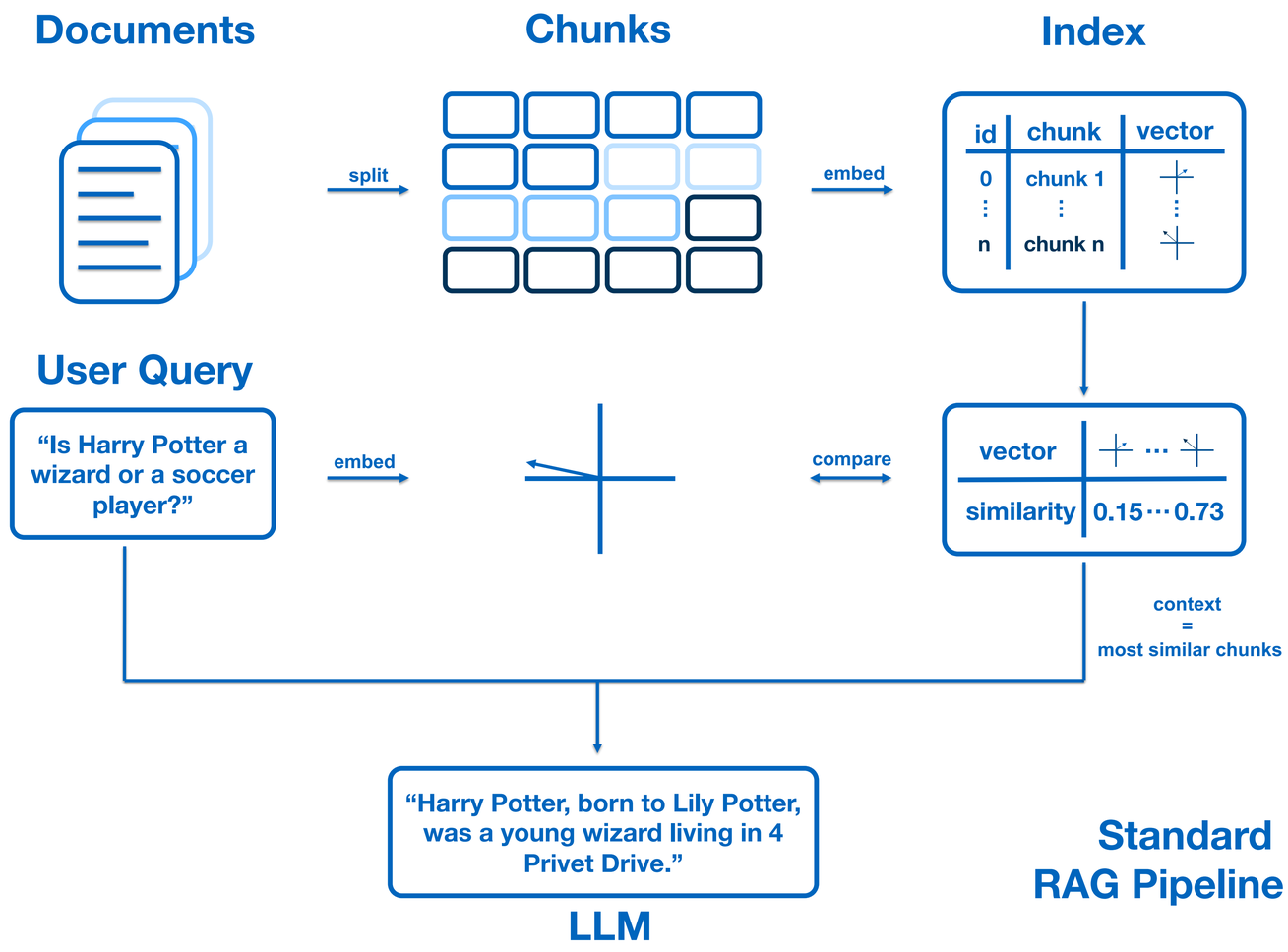
Hamdi Berke Göçmen, Guangyao Quan, Luca Mattes Wiehe
Supervisor: Miriam Anschütz (Social Computing Group)

Motivation

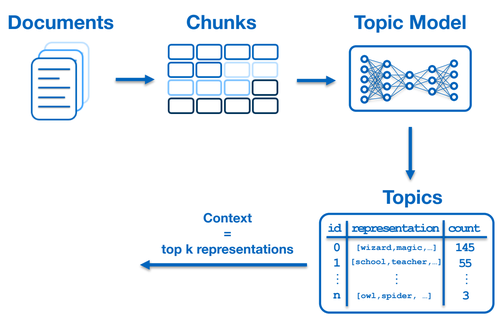
1. **LLMs are prone to hallucination**
   - Reason 1: Query too short (not enough context)
   - Reason 2: Query too long (assessment of relevance is difficult)
2. **Retrieval Augment Generation (RAG) helps**
   - Create an index from trustworthy documents
   - Use the query to obtain most relevant chunks
   - Pass chunks to LLM to add context and help prioritize information
3. **Standard RAG not optimized for summarization**
   - Summarization queries don't give rise to specific chunks
   - Obtained chunks will not represent important information
   - Summarization uses information from many chunks

Idea

1. **Adjust RAG Pipeline for summarization**
   - The user query does not point to specific chunks. Are there alternative ways to prioritize chunks in a summarization setting?
   - For summarization, many chunks can contain relevant information. Why don't we extract key concepts from the source text and pass them directly?
2. **Do adjustments improve summarization?**
   - **Baseline 1: Standard RAG Pipeline**
     Directly compare query to full chunks
   - **Baseline 2: Sentence-Window Retrieval**
     Compare query to sentences instead of full chunks => more detailed
   - **Baseline 3: Auto-Merging Retrieval**
     Compare query to chunks in a hierarchical fashion => more diverse

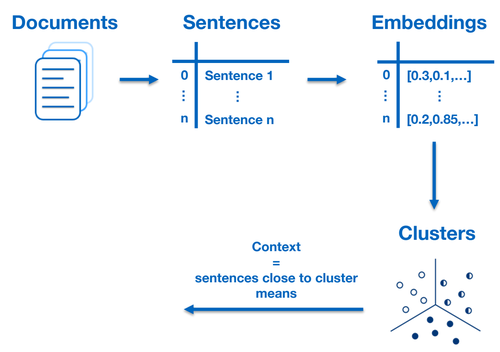


- Idea: Most frequent topic = relevant for summary
- Use topic model to extract topic for each chunk
- Pass summaries of *top k* topics as context



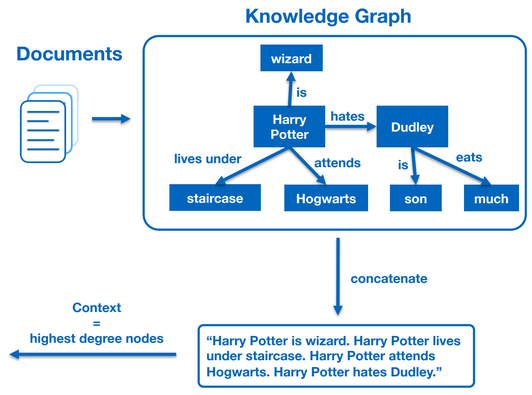
Approach 1: Topic-based Retrieval

- Idea: Cluster centers represent different ideas
- Cluster sentences based on their embeddings
- Return sentences close to cluster centers



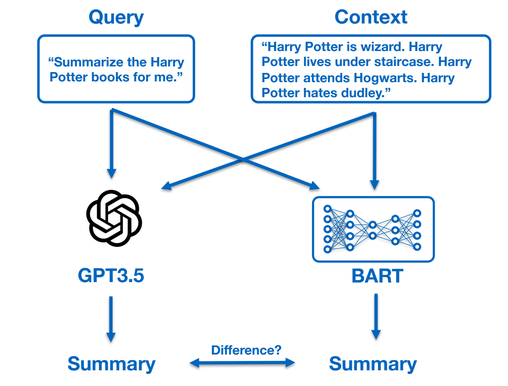
Approach 2: Extractive Summarization

- Idea: Many relationships = important word
- Create knowledge graph from source text
- Concatenate all relationships and use as context



Approach 3: Knowledge-Graph Retrieval

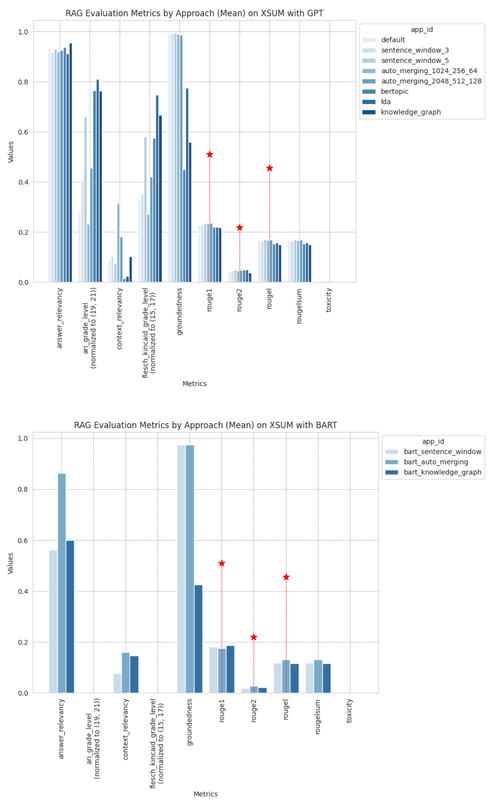
- Idea: Find out how much choice of LLM matters
- Use a different LLM than GPT3.5 Turbo
- Reduce API costs by using BART as a local LLM

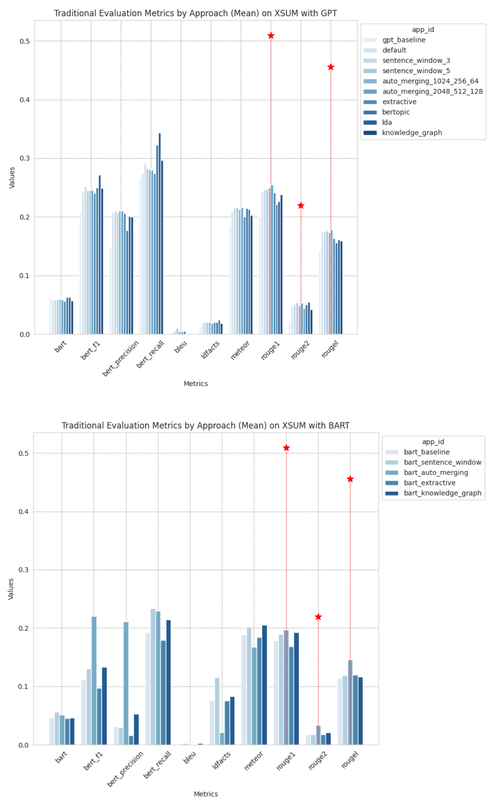


Approach 4: Using BART as LLM

Results

- GPT3.5 Turbo outperforms BART
- Our approaches perform better than Standard RAG
- Retrieved Context still not perfect
- Low toxicity, high answer relevance and groundedness for all approaches
- Low BLEU score because of differing summary lengths
- Low factuality across approaches





Conclusion

1. All RAG approaches (including baseline) have very similar performance.
2. LLM capabilities are most decisive for quality.
3. Knowledge Graph Retrieval consistently outperforms the other approaches in terms of summarization quality.
4. Disadvantage: Knowledge Graph takes long time to build, in particular for larger documents

Future Work

1. Experiments with Hyperparameters: Chunk Size, Extraction Ratio, Prompt Optimization.
2. How much improvement can RAG achieve, when dealing with multi-document summaries?

Technische Universität München
TUM School of Computation, Information and Technology
Research Group Social Computing

Poster for Master Lab Course - Machine Learning for Natural Language Processing Applications (IN2106, IN4249)