Retrieval Augmented Generation

Assignment 3

Information Retrieval Course

Academic Year 2025/2026

Group Members:

Alperen Davran	s0250946
Matteo Carlo Comi	s0259766
Shakhzodbek Bakhtiyorov	s0242661
Amin Borqal	s0259707

University of Antwerp
Faculty of Science
Department of Computer Science

January 30, 2026

GitHub Repository:
https://github.com/alperendavran/Information-Retrieval-Assignment-3

Contents

1 Introduction

Retrieval Augmented Generation (RAG) systems combine information retrieval with large language models to provide accurate, grounded answers to user queries. This project implements a RAG system for the University of Antwerp Computer Science Masters program, addressing the challenge of providing prospective and current students with accurate information about courses, prerequisites, schedules, and program requirements.

1.1 Motivation

Traditional question-answering systems often struggle with domain-specific queries or produce hallucinated responses when relying solely on the parametric knowledge of language models. RAG systems address these limitations by:

- Grounding responses in retrieved source documents
- Reducing hallucinations through explicit context
- Enabling transparent attribution of information sources
- Supporting scalable knowledge updates without model retraining

1.2 System Overview

Our implementation consists of five core components:

1. **Document Chunking**: Splitting course descriptions into semantically coherent passages
2. **Embedding & Indexing**: Computing dense vector representations and building efficient retrieval indices
3. **Retrieval Module**: Encoding queries and retrieving top-k similar passages
4. **Answer Generation**: Using GPT-4o to synthesize answers from retrieved context
5. **Evaluation Framework**: Assessing retrieval quality, answer quality, and error patterns

We implement two system variants:

- **Baseline**: Standard dense retrieval with top-k passage selection
- **Agentic**: Advanced workflow with multi-query expansion and reciprocal rank fusion

2 System Architecture

2.1 Document Chunking (10%)

2.1.1 Dataset

The dataset consists of scraped webpages from the University of Antwerp Computer Science Masters program, including:

- Course descriptions (prerequisites, learning outcomes, content, assessment)
- Program structure information
- Study programme overviews for three tracks: Software Engineering, Data Science & AI, and Computer Networks

After preprocessing, the dataset contains **588 passages** across all course materials.

2.1.2 Chunking Strategy

Chunk Size: 250 tokens (approximately 180-220 words)

Rationale:

- Large enough to capture complete semantic units (e.g., a course's prerequisites section)
- Small enough to maintain focused relevance for specific queries
- Balances context window efficiency in the generation model
- Aligns with typical paragraph/section lengths in course descriptions

Overlap: 15% (approximately 37 tokens)

This overlap ensures:

- Important information at chunk boundaries is not lost
- Improved retrieval recall for queries spanning multiple chunks
- Better semantic continuity across passages

2.1.3 Preprocessing

The following preprocessing steps were applied:

1. **HTML/Markdown removal**: Already performed in the provided dataset
2. **Text normalization**: Unicode normalization to handle special characters
3. **Sentence boundary preservation**: Chunks are split at sentence boundaries when possible to maintain semantic coherence

4. **Metadata extraction**: Each chunk retains metadata including course title, section type, and source URL

No lowercasing was applied, as:

- Course codes (e.g., "2500WETINT") are case-sensitive
- Proper nouns (e.g., "Miguel Camelo", "University of Antwerp") carry important information
- Modern embedding models handle case variations effectively

2.2 Embedding & Indexing (20%)

2.2.1 Embedding Model Selection

Model: `sentence-transformers/all-MiniLM-L6-v2`

Specifications:

- Embedding dimension: 384
- Max sequence length: 256 tokens
- Model size: 80MB
- Inference speed: 2,800 sentences/second on CPU

Justification:

1. **Local execution**: Runs efficiently on consumer hardware without GPU requirements
2. **Quality**: Achieves strong performance on semantic textual similarity benchmarks (Sentence Transformers leaderboard)
3. **Efficiency**: Small model size enables fast embedding computation
4. **Domain suitability**: Pre-trained on diverse text, generalizes well to educational content
5. **Community adoption**: Widely used baseline in RAG systems, enabling comparison with existing work

2.2.2 Indexing Strategy

Index Type: FAISS Flat Index (Inner Product)

Implementation Details:

- All embeddings are L2-normalized before indexing
- Inner product on normalized vectors is equivalent to cosine similarity
- Exact search guarantees optimal retrieval quality

- Index size: 588 vectors × 384 dimensions = 900KB (negligible memory footprint)

Rationale:

- Dataset size (588 passages) is small enough for exact search
- No need for approximate methods (e.g., IVF, HNSW) at this scale
- Exact search ensures reproducible evaluation results
- Simplifies debugging and error analysis

For larger datasets (>100K passages), we would consider:

- IVF (Inverted File) index for efficient clustering
- HNSW (Hierarchical Navigable Small World) for graph-based approximate search
- Product quantization for memory reduction

2.3 Retrieval Module (20%)

2.3.1 Query Processing

The retrieval pipeline consists of:

1. **Query encoding**: Same embedding model as documents (`all-MiniLM-L6-v2`)
2. **Normalization**: L2-normalize query embedding
3. **Similarity search**: FAISS inner product search
4. **Ranking**: Sort by similarity score (higher = more relevant)

2.3.2 Top-k Selection

k = 5 passages per query

Rationale:

- Balances recall (capturing relevant information) and precision (avoiding noise)
- Fits comfortably within GPT-4o's context window
- Empirically optimal for this dataset size based on preliminary experiments
- k=1 is too restrictive (low recall), k=20 introduces excessive noise

2.3.3 Agentic Retrieval Enhancements

The agentic system extends baseline retrieval with:

1. **Multi-query expansion**: Generate 2-3 query variations
2. **Reciprocal Rank Fusion (RRF)**: Merge results from multiple queries
3. **Post-retrieval deduplication**: Remove duplicate passages
4. **MMR diversification**: Maximal Marginal Relevance for result diversity

RRF Formula:

$$RRF(d) = \sum_{q \in Q} \frac{1}{k + rank_q(d)} \tag{1}$$

where $k = 60$ is a smoothing parameter, Q is the set of query variations, and $rank_q(d)$ is the rank of document d for query q.

2.4 Answer Generation Module (30%)

2.4.1 Generation Pipeline

1. **Context assembly**: Concatenate top-k passages with metadata
2. **Prompt construction**: Insert context and query into structured prompt
3. **LLM inference**: GPT-4o generates grounded answer
4. **Source attribution**: Return passages with similarity scores

2.4.2 Prompt Template

The system prompt instructs GPT-4o to:

- Answer based **only** on provided context
- Cite source passages when possible
- Explicitly state "I don't have enough information" if context is insufficient
- Avoid speculation or hallucination
- Maintain factual accuracy

Example Prompt Structure:

```
You are a helpful assistant for the University of
Antwerp Computer Science Masters program. Answer the
question based ONLY on the provided context.

Context:
```

```
[Passage 1]: ...
[Passage 2]: ...
...

Question: {query}

Answer:
```

2.4.3 Generation Parameters

- **Model**: GPT-4o (`gpt-4o`)
- **Temperature**: 0.1 (low temperature for factual responses)
- **Max tokens**: 1000 (sufficient for detailed answers)
- **API configuration**: Uses provided API key with rate limiting

2.4.4 Baseline Comparison

We compare RAG answers to **baseline answers** (GPT-4o without retrieval):

Example Comparison:

Query: "How many ECTS is Internet of Things?"

WITH RETRIEVAL:

> "The Internet of Things course is worth 6 ECTS credits."

WITHOUT RETRIEVAL (baseline):

> "The 'Internet of Things' course in the University of Antwerp's Computer Science Master's program is typically worth 6 ECTS credits. However, it's always a good idea to check the most current course catalog..."

Observations:

- RAG provides concise, definitive answer
- Baseline hedges with uncertainty despite correct answer
- RAG includes source attribution

3 System Evaluation (20%)

3.1 Evaluation Methodology

We employ a multi-faceted evaluation approach:

3.1.1 Pooled Relevance Judgments

1. **Query set**: 19 test queries covering diverse information needs
2. **Candidate pooling**: Combine top-20 results from both systems
3. **LLM-based labeling**: GPT-4o-mini judges relevance using function calling
4. **Metrics**: Recall@5, MRR, nDCG@5, MAP

Why pooled evaluation?

- Avoids bias toward individual systems
- Provides fair comparison on shared relevance labels
- Enables reliable ablation analysis

3.1.2 Cost Analysis

Total evaluation cost: **$0.064** (6.4 cents)

Operation	Events	Cost (USD)	%
LLM label relevance	322	0.035297	55.1%
LLM judge faithfulness	78	0.015455	24.1%
Answer generation	65	0.010682	16.7%
LLM judge compare	20	0.002531	4.0%
Tool router	1	0.000096	0.1%
Total	**486**	**0.064062**	**100%**

Table 1: OpenAI API cost breakdown (model: gpt-4o-mini)

3.2 Retrieval Quality

3.2.1 Quantitative Results

System	Recall@5	MRR	nDCG@5	MAP
Baseline	0.796±0.284	1.000±0.000	0.958±0.137	0.950±0.132
Agentic	0.730±0.303	1.000±0.000	0.902±0.189	0.974±0.078

Table 2: Retrieval quality on pooled relevance labels (n=19 queries)

Key Findings:

- **Baseline outperforms agentic** on Recall@5 (0.796 vs 0.730)
- Both systems achieve **perfect MRR** (1.0), indicating top-1 results are highly relevant

- **High standard deviations** reflect query difficulty variation
- Agentic system shows **slightly higher MAP** (0.974 vs 0.950), suggesting better overall ranking

3.2.2 Interpretation

The **baseline's superior Recall@5** is surprising given the agentic system's query expansion. Possible explanations:

1. **Query drift**: Expanded queries introduce noise, retrieving less relevant passages
2. **Rank fusion artifacts**: RRF may down-weight highly relevant passages from original query
3. **Dataset size**: At 588 passages, sophisticated retrieval may be unnecessary
4. **Evaluation bias**: Pooled labels may favor simpler retrieval strategies

Perfect MRR indicates:

- Most queries have at least one highly relevant passage in top-1
- Embedding model effectively captures semantic similarity
- Query formulation in test set is clear and unambiguous

3.3 Answer Quality

3.3.1 Manual Inspection Examples

Example 1: Factual Query

Query: "How many ECTS is Internet of Things?"
Retrieved passages: 5 highly relevant chunks from IoT course description
Generated answer:

> "The Internet of Things course is worth 6 ECTS credits."

Evaluation:

- **Correctness**: (verified against ground truth)
- **Completeness**: (answers fully)
- **Hallucination**: None
- **Source attribution**: (cites IoT course page)

Example 2: Complex Multi-Part Query

Query: "Who teaches Internet of Things and what semester is it in?"
WITH RETRIEVAL:

"The course 'Internet of Things' is taught by Miguel Camelo and is offered in the 2nd semester (2E SEM)."

WITHOUT RETRIEVAL:

"The course 'Internet of Things' in the University of Antwerp's Computer Science Master's program is taught by Peter Hellinckx. It is offered in the first semester."

Evaluation:

- RAG: **Correct** (Miguel Camelo, 2nd semester)
- Baseline: **Incorrect** (wrong instructor, wrong semester)
- Demonstrates critical value of retrieval for factual accuracy

Example 3: Insufficient Context
Query: "What programming languages do I need for IoT?"
Retrieved passages: General IoT prerequisites mentioning "C, C++, Python"
Generated answer:

"I don't have enough information in the provided context to answer this question."

Analysis:

- System correctly identifies insufficient detail
- Prerequisites mention languages but not specific proficiency requirements
- Conservative response avoids hallucination

3.3.2 Faithfulness Analysis

We use GPT-4o-mini as a judge to evaluate **faithfulness** (does the answer match the retrieved context?):

Judgment	Count	%
Faithful	61	78.2%
Partially faithful	12	15.4%
Unfaithful	5	6.4%
Total	**78**	**100%**

Table 3: Faithfulness evaluation (LLM-as-judge)

Observations:

- 78.2% of answers are fully faithful to retrieved context
- Low temperature (0.1) reduces hallucination risk
- Unfaithful cases often involve inference beyond provided facts

3.4 Error Analysis

3.4.1 Retrieval Failures (3 cases)

Case 1: Second Semester Compulsory Courses (Data Science)

Query: "In the Data Science study programme (2025-2026), which compulsory courses are in the 2nd semester?"

Recall@5: 0.250 (5 relevant passages retrieved, 20 total relevant)

Issue:

- Query requires **aggregating information** from 20 separate course descriptions
- Each course is in a separate chunk
- Top-5 retrieval fundamentally insufficient for comprehensive answer
- Embedding similarity favors individual courses over complete program structure

Retrieved (relevant):

1. Current Trends in DS & AI (2000WETCTD, 2E SEM)
2. Algorithmic Foundations of DS (2001WETATD, 2E SEM)
3. Bioinformatics (2002WETBIN, 2E SEM)
4. Nonsmooth Optimisation (2600WETNOO, 2E SEM)
5. Software Reengineering (2001WETSRE, 2E SEM)

Missed (relevant):

- Internet of Things, Master Thesis, Programming Paradigms, Data Mining, Software Testing, Neural Networks, Research Project, Internship, Case Studies, Convex Analysis (15 more courses)

Root cause:

- **Chunk granularity mismatch**: Individual courses are atomic units, but query needs course *list*
- **Embedding bias**: Similarity model ranks specific course matches over program-level information
- **k-value limitation**: Even k=20 would miss some courses

Potential solutions:

1. Create **summary chunks** listing all courses per semester/program
2. Implement **hierarchical retrieval** (program → semester → individual courses)

3. Use **query expansion** to explicitly request course lists
4. Increase k or use **multi-hop retrieval**

Case 2: Instructor-Specific Query (Computer Networks)

Query: "Which compulsory course in the Computer Networks study programme is taught by Juan Felipe Botero, and what is its exam period?"

Recall@5: 0.250 (1 relevant out of 4 total)

Retrieved:

1. Future Internet (2500WETFUI, Juan Felipe Botero) **Relevant**
2. Specification & Verification (different instructor)
3. Programming Paradigms (different instructor)
4. Topics in Computer Networks (different instructor)
5. Internet of Things (different instructor)

Issue:

- Query contains **specific instructor name** (Juan Felipe Botero)
- Embedding model struggles with **named entity matching**
- Retrieved passages mention course names but not instructors prominently
- Metadata (instructor field) not leveraged in retrieval

Root cause:

- Dense embeddings capture semantic similarity, not exact entity matches
- Instructor names appear as metadata, not in passage text
- Lack of **hybrid retrieval** (dense + sparse/keyword)

Potential solutions:

1. **Hybrid retrieval**: Combine dense embeddings with BM25 for keyword matching
2. **Metadata filtering**: Use structured filters (e.g., "instructor=Juan Felipe Botero")
3. **Entity-aware chunking**: Include instructor names in passage text
4. **Query rewriting**: Extract entities and create targeted sub-queries

Case 3: First Semester Course List (Computer Networks)

Query: "In the Computer Networks study programme (2025-2026), list the compulsory courses in the 1st semester (names + course codes)."

Recall@5: 0.312 (5 relevant out of 16 total)

Retrieved (all relevant):

1. Specification & Verification (2001WETFSP, 1E SEM)
2. Data Science & Ethics (2104TEWDAS, 1E SEM)
3. Future Internet (2500WETFUI, 1E SEM)
4. AI Project (2001WETPDB, 1E SEM)
5. Optimisation (1601WETOPT, 1E SEM)

Missed (11 courses): Model Driven Engineering, Numerical Optimisation, Database Systems, Academic Literacies, Biomedical AI, Advanced Wireless & 5G, Information Retrieval, Math Foundations of RL, Modelling of Software Systems, etc.

Issue:

- Similar to Case 1: **aggregation problem**
- Query needs 16 courses, k=5 is insufficient
- No single chunk contains complete course list

Why Recall@5 > Case 1?

- Fewer total relevant passages (16 vs 20)
- First semester courses may have stronger embedding similarity to query
- Random variation in ranking

3.4.2 Generation Errors (3 cases)

Case 1: Hallucinated Instructor (Baseline)

Query: "Who teaches Internet of Things and what semester is it in?"

WITHOUT RETRIEVAL (Baseline):

> "The course 'Internet of Things' in the University of Antwerp's Computer Science Master's program is taught by **Peter Hellinckx**. It is offered in the **first semester**."

Ground Truth:

- Instructor: Miguel Camelo
- Semester: 2E SEM (second semester)

Analysis:

- GPT-4o's parametric knowledge is **outdated** (knowledge cutoff: January 2025)
- Peter Hellinckx may have taught IoT in previous years
- Without retrieval, model defaults to plausible but incorrect information

- Demonstrates **critical value of RAG** for time-sensitive facts

Case 2: Over-Generalization (WITH RETRIEVAL)
Query: "Can I use AI tools in my Master thesis?"
Retrieved context (from Master Thesis assessment criteria):

> "Students may use generative AI tools for writing the master thesis, similar to initial search engines such as Google and for checking grammar and spelling. During the thesis defence, students may be asked to elaborate on how they used generative AI."

Generated answer:

> "Yes, you can use generative AI tools in your Master thesis. You may use them for writing the thesis, similar to using search engines like Google, and for checking grammar and spelling. However, it must be clearly indicated which code was created by which tools, and during the thesis defence, you may be asked to elaborate on how you used generative AI."

Issue:

- Answer conflates **two separate contexts**: writing assistance and code generation
- "which code was created by which tools" comes from a different section (assessment of code projects)
- Not technically a hallucination, but **imprecise context merging**

Root cause:

- Multiple passages discuss AI tool usage in different contexts
- Generation model struggles to disambiguate thesis *writing* vs thesis *code*
- Prompt doesn't enforce strict passage boundaries

Case 3: Insufficient Context Handling
Query: "What programming languages do I need for IoT?"
Retrieved context:

> "Prerequisites: Programming in C, C++, Python (1Ba course 'Inleiding Programmeren' or equivalent)"

Generated answer:

> "I don't have enough information in the provided context to answer this question."

Analysis:

- Answer is **technically correct** (prerequisites specific language requirements)

- However, a more helpful response would acknowledge C/C++/Python are mentioned as prerequisites
- System prompt prioritizes **conservatism over helpfulness**

Trade-off:

- **Conservative**: Reduces hallucination risk, but may under-utilize available information
- **Aggressive**: Provides more complete answers, but risks inference beyond facts

Baseline (WITHOUT RETRIEVAL) provides detailed answer:

> "For the Internet of Things (IoT), several programming languages are commonly used... C/C++, Python, Java, JavaScript, Rust, Lua, Go, Swift..."

Comparison:

- Baseline is more informative but **not grounded** in UAntwerp's specific requirements
- RAG answer is **safer but less helpful**
- Highlights importance of prompt engineering for balancing precision/recall

3.5 Summary of Findings

Component	Key Insight
Chunking	250-token chunks with 15% overlap balance semantic coherence and retrieval precision
Embedding	all-MiniLM-L6-v2 provides strong quality-efficiency trade-off for local deployment
Indexing	Flat index sufficient for 588 passages; exact search ensures reproducibility
Retrieval	Baseline outperforms agentic (Recall@5: 0.796 vs 0.730); query expansion may introduce noise
Generation	RAG reduces hallucinations vs baseline; low temperature (0.1) improves faithfulness
Errors	Aggregation queries, entity matching, and conservative prompts are key failure modes

Table 4: Summary of evaluation findings

4 Discussion

4.1 System Performance

4.1.1 Strengths

1. **High retrieval quality**: Baseline achieves 79.6% Recall@5, with perfect MRR

2. **Factual accuracy**: RAG significantly reduces hallucinations vs baseline LLM
3. **Efficient architecture**: Local embeddings + exact search enable fast inference
4. **Low cost**: Full evaluation costs only $0.064 using GPT-4o-mini
5. **Transparent attribution**: Source passages enable verification

4.1.2 Limitations

1. **Aggregation queries**: Cannot answer questions requiring 15+ passages
2. **Entity matching**: Struggles with specific names, course codes in queries
3. **Agentic underperformance**: Query expansion introduces noise at this scale
4. **Conservative generation**: May under-utilize available context
5. **No temporal awareness**: Cannot distinguish current vs historical information

4.2 Baseline vs Agentic Trade-offs

Dimension	Baseline	Agentic
Recall@5	**0.796±0.284**	0.730±0.303
MRR	1.000±0.000	1.000±0.000
nDCG@5	**0.958±0.137**	0.902±0.189
MAP	0.950±0.132	**0.974±0.078**
Complexity	Low	High
Latency	**Fast**	Slow (3x queries)
Robustness	**High**	Sensitive to expansion

Table 5: Baseline vs Agentic comparison

Recommendation: For this dataset size (588 passages), **baseline is preferable**. Agentic techniques may benefit larger, noisier corpora.

5 Conclusion

This project successfully implemented a complete RAG system for the University of Antwerp Computer Science Masters program. Key achievements include:

- **Effective chunking strategy**: 250 tokens with 15% overlap balances coherence and granularity
- **Efficient local deployment**: all-MiniLM-L6-v2 + FAISS enables fast, reproducible retrieval
- **Strong baseline performance**: 79.6% Recall@5 with perfect MRR on pooled labels

- **Reduced hallucinations**: RAG provides factually grounded answers vs baseline LLM
- **Comprehensive evaluation**: Multi-dimensional assessment (retrieval, generation, errors)

Our error analysis reveals critical insights:

1. **Aggregation queries** (e.g., "list all courses") challenge fixed-k retrieval
2. **Entity matching** (e.g., instructor names) requires hybrid retrieval
3. **Conservative prompts** reduce hallucinations but may under-utilize context

Surprisingly, the **baseline outperforms the agentic system** at this scale. Query expansion introduces noise when the corpus is small and well-structured. This highlights the importance of **matching system complexity to dataset characteristics**.

For production deployment, we recommend:

- Hybrid retrieval (dense + sparse) for entity-rich queries
- Hierarchical indexing for aggregation tasks
- User feedback loops to refine relevance judgments
- Continuous evaluation as corpus evolves

Overall, this RAG system demonstrates the viability of retrieval-augmented generation for domain-specific question answering, achieving strong performance with minimal computational overhead.

References

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.

[2] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. T. (2020). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

[3] Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2024). RAGAS: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*.

[4] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *arXiv preprint arXiv:1908.10084*.

[5] Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535-547.

6 Appendix: Sample Queries and Outputs

This appendix contains the complete set of test queries used in our evaluation.

6.1 Test Query 1: Factual Detail

Query: "How many ECTS is Internet of Things?"
With RAG: "The Internet of Things course is worth 6 ECTS credits."
Without RAG: "The 'Internet of Things' course in the University of Antwerp's Computer Science Master's program is typically worth 6 ECTS credits. However, it's always a good idea to check the most current course catalog..."
Result: Both correct, but RAG is more confident and direct.

6.2 Test Query 2: Missing Information

Query: "What programming languages do I need for IoT?"
With RAG: "I don't have enough information in the provided context to answer this question."
Without RAG: "For the Internet of Things (IoT), several programming languages are commonly used: C/C++, Python, Java, JavaScript, Rust, Lua, Go, Swift..."
Result: RAG correctly refuses (though information was in dataset but not retrieved). Baseline provides generic answer.

6.3 Test Query 3: Policy Question

Query: "Can I use AI tools in my Master thesis?"
With RAG: "Yes, you can use generative AI tools in your Master thesis. You may use them for writing the thesis, similar to using search engines like Google, and for checking grammar and spelling. However, it must be clearly indicated which code was created by which tools, and during the thesis defence, you may be asked to elaborate on how you used generative AI."
Without RAG: "The use of AI tools in your Master thesis at the University of Antwerp generally depends on the guidelines set by the university and your specific department. Ensure compliance with academic integrity policies..."
Result: RAG provides specific UAntwerp policy. Baseline gives generic advice.

6.4 Test Query 4: Multi-Part Query

Query: "Who teaches Internet of Things and what semester is it in?"
With RAG: "The course 'Internet of Things' is taught by Miguel Camelo and is offered in the 2nd semester (2E SEM)."
Without RAG: "The course 'Internet of Things' in the University of Antwerp's Computer Science Master's program is taught by Peter Hellinckx. It is offered in the first semester."

Result: RAG 100% correct. Baseline hallucinated both facts (wrong teacher, wrong semester).