Developing and Benchmarking an Agentic Rag Approach Based on Conversational Datasets

IDP Report - TUM Supervisors: Andreas Finkenzeller, Prof. Dr. Sebastian Steinhorst - Mitra Labs Supervisor: Alexander Wiener

Nicolas Stellwag
Technical University of Munich
Mitra Labs
nicolas.stellwag@tum.de

***Abstract*—Retrieval-Augmented Generation (RAG) systems enhance Large Language Models (LLMs) but often lack the ability to perform queries beyond simple semantic retrieval. Furthermore, they are not built with the requirement to run locally on consumer hardware. We tackle this niche by introducing Coco, an LLM agent designed to help its users reflect on transcriptions of their conversations. Coco's tools allow it to perform multi-step reasoning, metadata filtering (time, speakers), and retrieval based on language emotion. For evaluation, we present a synthetic benchmark dataset with 100 question-answer pairs covering five key capabilities. Experiments comparing Coco (using Llama-3.3-70B-Instruct and gpt-4o-2024-11-20) against traditional RAG show the agentic approach offers significant advantages. While the Llama-based agent achieves a 7.4% improvement in GEval Correctness over its RAG counterpart, the 4o agent widened this gap to 31.8%, demonstrating the potential of agentic RAG while highlighting the current bottleneck imposed by the base model's tool calling capability. Code and dataset are publicly available[1].**

I. Introduction

A. Motivation

Large Language Models (LLMs) and *Retrieval Augmented Generation* (RAG) have established themselves as powerful tools in a wide variety of text-based domains. But services and products built around them mostly focus on productivity related tasks such as programming or customer service. Mitra Labs aims at leveraging the capabilities of modern *Natural Language Processing* (NLP) technology to build *Coco*, a knowledge-enhanced chatbot with access to a database of transcribed conversations. Coco can help its user to reflect on his life, track progress with respect to goals, and identify patterns of behavior in conversations.

B. Problem Statement

This problem implies a set of technical requirements. Firstly, Coco needs to be able to aggregate and abstract information of its knowledge base. For instance, it should be able to answer a question like "What were my New Year's resolutions and am I sticking to them?", which requires multiple queries to a knowledge base and summarization of the retrieved information. Secondly, Coco should be able to take meta-information like temporal information into account. An example would be the question "What did I talk to Alice about last week?", which requires constraining the considered transcripts to those from recordings of last week. Finally, due to the sensitivity of the data, Coco should be able to run locally on consumer-grade hardware. Consequently, available compute and memory are heavily constrained.

Existing approaches might tackle individual aspects of these requirements, but there is no "all-in-one" solution available. That also makes quantifying Coco's performance a hard task, because no public benchmark fits our use case.

C. Contributions

Our contributions are two-fold:

- We introduce Coco, an LLM-agent with a set of tools to query its knowledge base in a more sophisticated manner than current RAG approaches.
- To quantify Coco's performance, we introduce a synthetic benchmark dataset comprising 500 chunks and 100 samples that cover 5 different capabilities.

II. Background

A. Text Embeddings

Generally, text embeddings are a numerical vector representations of text that encode its semantic meaning. That allows to perform a range of text-based operations as numerical operations. Most importantly, semantic similarity of text can be approximated as vector similarity of the respective embeddings.

Contextual token embeddings are vector representations of a single *token*, which can be a single character, a part of a word, a full word, or even multiple words. Which and how many characters are fused into a token is usually determined by statistics about their co-occurence in a text corpus [1]. We make use of contextual token embeddings to compute metrics that quantify the similarity of the ground truth answers of our benchmark and Coco's predicted answers.

Modern contextual token embeddings are generated by *text encoder* neural networks. The majority of them are derived from a model called *BERT* [2], which is why text encoders are often referred to as BERT-style models. To compute the embeddings, a sentence S is first split to tokens $S =$

[1] https://github.com/mitralabs/coco

$(t_1, \ldots, t_n) \in V^n$. Afterward, the whole sequence of tokens is passed to a BERT-style encoder as

$$(h_{\text{CLS}}, h_1, \ldots, h_n, h_{\text{SEP}}) = \text{BERT}\Big(([\text{CLS}], t_1, \ldots, t_n, [\text{SEP}])\Big) \quad (1)$$

where [CLS] is a special token that will become relevant in the subsequent paragraph and [SEP] is a special token that signals the end of a sentence to the model. Since all tokens of a sentence are processed together, each token's embedding takes its context into account. For instance, "mouse" would get assigned different embeddings depending on whether it occurs in the text "I bought a mouse for my PC" or "My cat caught a mouse".

BERT-style models obtain their extensive understanding of language from being trained on proxy-tasks on massive unlabeled text corpora. The original BERT was trained on a corpus of roughly 3.3 billion words on *Masked Language Modeling* (MLM) [2], which means predicting randomly masked words of the sequence. That language understanding can also be leveraged to generate embeddings that encode the meaning of the full sentence or text sequence.

Sentence embeddings encode the semantic meaning of a sequence of tokens. (That sequence can be an actual sentence or multiple sentences, the name is established but can be misleading.) The starting point for a sentence embedding model is a pretrained BERT-style text encoder. The sentence embedding is then computed as the contextual token embedding of the special [CLS] token. Due to the nature of the pretraining task, this embedding is no meaningful representation of the whole sequence's semantics yet. That's why the pretrained BERT models are finetuned on sentence-level tasks.

For instance, if the sentence embeddings are used to retrieve texts that are semantically similar to a query text, a very simple finetuning procedure would look as follows: *1)* Obtain a dataset with labeled positive sentence pairs (same semantic meaning) and create negative pairs (different semantic meaning) as random pairs. *2)* Compute [CLS] embeddings for both sentences of the pairs. *3)* Compute cosine similarities between both embeddings as

$$\text{cos\_sim}(a, b) = \frac{a \cdot b}{||a||_2 \cdot ||b||_2} \quad (2)$$

for each pair. *4)* Pull the cosine similarities of positive pairs towards 1 and of negative pairs to -1. Since the model already has learned useful intermediate representations for all tokens during pretraining, it now merely has to learn how to aggregate them into the [CLS] embedding such that the embedding's direction encodes the full text's semantic meaning.

We use sentence embeddings to allow Coco to retrieve relevant knowledge from its database by performing a similarity search. For better retrieval performance, we use *BGE-M3* [3], a BERT-style text encoder finetuned with a more sophisticated method: *1)* For each positive pair, obtain hard negative sentences using the sampling method introduced by Xiong, Xiong, Li, *et al.* [4]. Hard negative sentences are semantically close to the sentences of the positive pair, but not similar. *2)* For one positive pair, and between sentences of the positive pair and the hard negative sentences, compute cosine similarities of their sentence embeddings. *3)* Constrain the sum of all similarities to 1 and pull the positive pair towards 1. That implicitly reduces similarities of pairs consisting of one positive and one hard negative sentence. Consequently, the model gets better at taking semantic subtleties into account, which makes it useful for semantic retrieval.

B. Large Language Models

LLMs are used for text generation. Their training task is given a sequence of text tokens, predict a discrete probability distribution over the whole token vocabulary for which token comes next. Text can then be generated by *autoregressive sampling*, which means repeatedly sampling the predicted distribution and appending the respective token to the sequence. Thus, LLMs' knowledge is constrained by the training corpus, and the LLM might even *hallucinate* wrong facts. As a result, vanilla LLMs are not useful as personalized chatbots because they cannot reference any facts about the individual user.

C. RAG

Retrieval Augmented Generation refers to a variety of techniques that first fetch context relevant to the user query, pass that knowledge to an LLM via the prompt, and only then return a final answer to the user. Usually, the prompt also contains an instruction to answer with "I don't know" if the user's question cannot be answered from the retrieved context. Hence, RAG is an efficient method to provide an LLM with individualized knowledge without retraining, and reducing hallucinations.

In the simplest and most widespread approach, referred to as *traditional RAG* from now on, that is achieved by fetching text chunks from a database, and passing the top k in terms of sentence embedding similarity to the user query as context to the LLM. This approach serves as the baseline we compare the Coco agent to.

D. LLM Agents

Artificial Intelligence (AI) agents were conceptualized long ago. Russell and Norvig [5] define agents as "Anything that perceives its environment and acts upon it". LLMs can certainly perceive their environment through text representations, but generating new text hardly qualifies as acting upon it. Schick, Dwivedi-Yu, Dessi, *et al.* [6] were the first to equip LLMs with the ability to call arbitrary functions of a programming language, thereby qualifying tool calling LLMs as agents. We leverage LLM agents for Coco to go beyond static RAG retrieval procedures by giving it the ability to perform arbitrary combinations of knowledge base queries via function calls.

III. Related Work

As previously mentioned, there is work related to ours. But existing work predominantly only deals with individual aspects of our requirements. Generally, the trend seems to be

going away from traditional RAG towards more sophisticated retrieval patterns that allow to answer complex questions.

There is a large body of work aimed at answering complex questions by performing *multi hop* retrieval, multiple semantic queries with different query sentences. *IRCoT* [7] does so by interleaving *chain of thought* (COT) steps with retrieval. COT means decomposing the user query to a set of steps and performing them sequentially. IRCoT always uses the current COT step as semantic query. *ITER-RETGEN* [8] tackles multi hop questions with iterative retrieval to refine the answer, always using the original query plus the generated answer as the new query. *RA-ISF* [9] integrates modules that can recursively break down complex questions into simpler subquestions when the system's confidence in the answer is low. Similarly, *MetaRAG* [10] employs a metacognitive component to dynamically rewrite retrieval queries and generation prompts if the initial answer quality is deemed insufficient. *RQ-RAG* [11] takes a different approach by fine-tuning an LLM specifically to decompose a user query into a structured tree of sub-queries for the RAG system. While we are faced with the same multi hop questions, these approaches are not feasible for us. They all employ rigid query patterns that are not flexible and general enough to be combined with solutions for our other requirements.

Other work deals with multi hop questions by giving an LLM agent the ability to perform semantic queries via tool calls. *Auto-RAG* [12] fine-tunes a very small tool-calling LLM agent on a synthetically generated dataset of RAG query chains. This is a much more general approach and similar to what Coco does. But since we also require strong performance on other tasks, a small, finetuned LLM is not feasible for us.

Besides answering multi hop questions, another challenge is incorporating metadata into the retrieval process. *Multi-MetaRAG* [13] addresses this by extracting metadata filters directly from the user query and using them to pre-filter the document database before semantic search. While we don't have a separate module only concerned with extracting metadata filters, our agent's retrieval tools also allow filtering considered pieces of knowledge.

IV. Dataset

In the following, we motivate and present the custom dataset we've created for benchmarking Coco.

A. Mitra Dataset

There is no public benchmark that tests all of Coco's capabilities. Existing benchmarks mostly cover simple semantic retrieval without multi hop requirements or metadata inclusion. Even though there are datasets testing single aspects of our requirements, they are not based on conversation excerpts.

Therefore, to test Coco properly, we present the *Mitra Dataset*, a public, synthetic, German, conversational dataset for retrieval and *question answering* (QA) tasks. It comprises a knowledge base consisting of 500 conversational chunks with recording timestamps and 100 samples consisting of a query, a ground truth answer, and the IDs of relevant chunks. The dataset is split into 5 different categories with 20 samples each, and each category is split into 15 training samples and 5 test samples.

To ensure an internally consistent knowledge base and correct samples, we leverage the large context window of "gemini-2.5-pro-exp-03-25" by Google DeepMind to create the initial version of the dataset in one shot. We ensure correctness and consistency for each individual category by asking the same LLM to verify the category's samples and chunks with respect to the following questions: *1)* Do the samples cover the use case of the respective category well? *2)* Are the ground truth answers correct? *3)* Is answering the ground truth answer possible without ambiguity, given the chunks? *4)* Do any ground truth answers contain information not explicitly required to answer the query question? Finally, we inspect random samples manually to further ensure correctness and consistency.

B. Categories

Each of the 5 categories is directly derived from one requirement for Coco. Examples for each category are presented in Table I.

Language Sentiment: Coco should be able to help the user on emotions expressed in certain situations. Accordingly, the system has to be able to fetch chunks by the emotions expressed in their language.

Multi Query: Coco should be able to answer complex questions based on its knowledge base. Therefore, the system must be able to perform multi hop retrieval.

People: Coco should be able to help the user track his relationships with specific people. Consequently, Coco must be able to filter conversations based on the people participating in them.

Summary: Coco should be able to help the user identify abstract patterns in his conversations. Hence, it should be able to summarize large quantities of chunks.

Time: Coco should be able to take temporal constraints into account. Thus, the system has to be able to filter chunks based on time, and interpret relative time statements of chunks in relation to the chunks' recording timestamps.

V. Coco Agent

In this section, we introduce the Coco agent and show how its tools give it all capabilities we require.

A. System Description

Coco is an agentic LLM with tools to perform RAG-like semantic queries. For each user query, Coco can decide to perform an arbitrary number of function calls before generating an answer. This makes our approach much more flexible than traditional RAG or even advanced RAG techniques that still rely on a static query pattern like those introduced in section III. Furthermore, additional capabilities are trivial to add by simply adding more tools. A flow diagram illustrating the difference between traditional RAG and the Coco agent can be found in Figure 1. Coco's tools are shown in Figure 2.

Figure 1. Flow diagram comparing traditional RAG to the Coco agent.

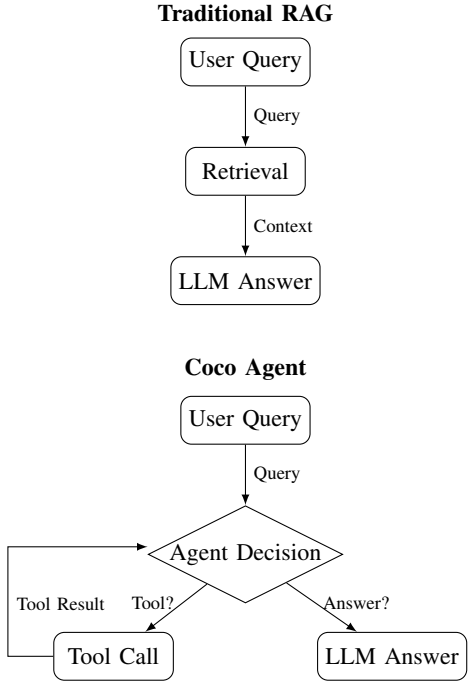

Figure 2. Tools available to the Coco agent.

procedure SEMANTIC_QUERY(query, [num], [start], [end], [substr])
  return Chunks via similarity between *query* embedding and chunk text embedding
  Filters: Applied before similarity search:
    If *start*: Filter for chunks after *start* time
    If *end*: Filter for chunks before *end* time
    If *substr*: Filter for chunks containing *substr*
  Returns: Top *num* (default 25) matching chunks (incl. content, metadata, distance)
end procedure

procedure EMOTION_QUERY(emotions, [num], [start], [end], [substr])
  return Chunks via similarity between *emotions* embedding and an embedding of a description of emotions present in the chunk language extracted using an LLM
  Filters: Applied before similarity search:
    If *start*: Filter for chunks after *start* time
    If *end*: Filter for chunks before *end* time
    If *substr*: Filter for chunks containing *substr*
  Returns: Top *num* (default 25) matching chunks (incl. content, metadata, distance)
end procedure

procedure GET_CURRENT_DATETIME
  return Current date and time (ISO format)
end procedure

B. Requirements

Coco can use its tools to tackle all required task categories (and therefore dataset categories) as follows:

Language Sentiment: Coco has the ability to retrieve chunks not just by semantic meaning, but also by emotions expressed in the chunk language using its `emotion_query` tool.

Multi Query: Coco can perform as many tool calls, and therefore as many `semantic_query` calls, as needed. Each decision for a next action is based on the whole conversation (and tool call) history, which means semantic queries can by dynamically issued based on the previous queries' results.

People: Coco has the ability to filter considered chunks based on the (case-insensitive) presence of a substring with both `semantic_query` and `emotion_query`. Since chunks are created with speaker diarization, that filter can be used to fetch chunks of conversations with a specific person.

Summary: Coco can perform multiple semantic queries and dynamically set the number of retrieved chunks.

Time: Coco can filter considered chunks for `semantic_query` and `emotion_query` based on time stamps. Its tools also return the recording timestamp metadata field in addition to the chunk text. So time references contained in chunks can be interpreted correctly.

VI. Evaluation

A. Metrics

We evaluate Coco's performance using 3 different classes of QA metrics. They all compare the generated final answer to the benchmark dataset's ground truth answer. *Metrics based on token n-grams* deal well with named entities and text that requires exact matches such as statements about time. Unfortunately, they do not take n-gram order into account and cannot deal well with synonyms, alternative formulations etc. *Embedding-based metrics* on the other hand capture semantic similarity well, independent of the exact word choice. But they have problems with negations and exact information such as dates and places, because different but similar information will still be very close in the embedding space. *LLM as a judge* metrics have the highest correlation with human judgement, but they are very expensive to compute and not fully deterministic.

BLEU [14] is an n-gram metric that primarily measures precision, indicating how much of the generated text is present in the ground truth. It is calculated as the geometric mean of modified n-gram precisions p_n for n-grams up to length N (in our case $N = 4$), multiplied by a brevity penalty BP to penalize prediction lengths c much shorter than the ground truth length r. SacreBLEU [15] provides a standardized implementation.

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3}$$

where $w_n = 1/N$ and the brevity penalty is:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{4}$$

The modified precision p_n is calculated as

$$p_n = \frac{\sum_{\text{n-gram} \in \text{prediction}} \text{Count}_{\text{clip}}(\text{n-gram})}{\sum_{\text{n-gram}' \in \text{prediction}'} \text{Count}(\text{n-gram}')} \tag{5}$$

where Count(n-gram) is the number of times an n-gram appears in the prediction, and $\text{Count}_{\text{clip}}$(n-gram) is the same count clipped to the maximum number of times that n-gram appears in any single ground truth text. This clipping prevents a prediction from getting high precision by repeating phrases with high frequency in natural language excessively. prediction is the set of all correctly predicted n-grams and prediction' is the set of all predicted n-grams. We employ BLEU to get a sense of Coco's precision with respect to exact information such as time.

ROUGE [16] is also an n-gram metric and focuses on recall, assessing how much of the ground truth text is captured in the prediction. ROUGE-N measures the recall of n-grams:

$$\text{ROUGE-N} = \frac{\sum_{\text{n-gram} \in \text{GT}} \text{Count}(\text{n-gram})}{\sum_{\text{n-gram}' \in \text{GT}'} \text{Count}(\text{n-gram}')} \tag{6}$$

where GT is the set of all correctly predicted n-grams and GT′ is the set of all predicted n-grams. ROUGE-L uses the Longest Common Subsequence (LCS) between a prediction Y (length n) and a ground truth X (length m) to compute recall R_{lcs}, precision P_{lcs}, and an F1-score F_{lcs}:

$$R_{\text{lcs}} = \frac{\text{LCS}(X, Y)}{m} \tag{7}$$

$$P_{\text{lcs}} = \frac{\text{LCS}(X, Y)}{n} \tag{8}$$

$$\text{ROUGE-L} = F_{\text{lcs}} = \frac{(1 + \beta^2) R_{\text{lcs}} P_{\text{lcs}}}{R_{\text{lcs}} + \beta^2 P_{\text{lcs}}} \tag{9}$$

In our case, $\beta = 1$. We use ROUGE to quantify Coco's recall with respect to exact information like time.

BERTScore [17] computes precision, recall, and F1 score based on the cosine similarity of contextual token embeddings (e.g., from BERT) between the prediction $(p_1, \ldots, p_m)$ and ground truth $(r_1, \ldots, r_k)$ texts.

$$\text{BERTScore R} = R_{\text{BERT}} = \frac{1}{k} \sum_{i=1}^{k} \max_{j=1,\ldots,m} \text{cos\_sim}(r_i, p_j) \tag{10}$$

$$\text{BERTScore P} = P_{\text{BERT}} = \frac{1}{m} \sum_{j=1}^{m} \max_{i=1,\ldots,k} \text{cos\_sim}(r_i, p_j) \tag{11}$$

$$\text{BERTScore F1} = F_{\text{BERT}} = \frac{2 P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{12}$$

We employ BERTScore to get a sense for precision and recall within individual samples in the semantic domain.

SemScore [18] measures the cosine similarity between the sentence embedding of the prediction E_{pred} and the ground truth answer E_{gt}, using a model explicitly fine-tuned for sentence similarity tasks.

$$\text{SemScore} = \text{cos\_sim}(E_{\text{pred}}, E_{\text{gt}}) \tag{13}$$

The model used by Aynetdinov and Akbik [18] is only trained on English text, which is why we additionally compute SemScore Multilingual using a strong sentence similarity model also trained on a German corpus called "paraphrase-multilingual-mpnet-base-v2". We use SemScore to quantify general semantic similarity between Coco's answers and the ground truth answers.

G-Eval [19] provides a framework for using LLMs as evaluators. Liu, Iter, Xu, *et al.* [19] define an evaluation criterion via prompt and let the LLM generate a COT to calculate it. We found that to be inconsistent and not correlate well with our judgement. A hard-coded sequence of steps works much better, ours can be found in Listing 1. While Liu, Iter, Xu, *et al.* [19] calculate their final score as the sum of all possible integer scores weighted by their predicted token probabilities, we have to use the score with maximum probability directly because we have to rely on an LLM API to compute the scores due to resource constraints. The LLM we employ as a judge is "Llama-3.3-70B-Instruct". While the metric is generated by an LLM and thus in principle not fully deterministic, we've found it to behave deterministically in practice with a temperature parameter of 0. This amplifies the differences in predicted probabilities so much, the predicted distribution approaches a discrete Dirac delta. We call our version GEval Correctness and find that it is the best heuristic for Coco's performance.

B. Setup

We compare the Coco agent against a traditional RAG baseline on our benchmark dataset. In both cases, we use "Llama-3.3-70B-Instruct" (Llama) because it is small enough to be deployable on consumer hardware, finetuned for tool calling, and available via public APIs for faster, parallel testing. The temperature is always set to 0 to make experiments (pseudo) deterministic. Our embedding model is BGE-M3 [3] in all cases because we found it to be the strongest compared to others we tried. For the traditional RAG, we use the top $k = 25$ most similar chunks to the user query as context. We keep intersecting sections of RAG prompt and agent system prompt as similar as possible. The RAG prompt template can be seen in Listing 2 and the Coco agent system prompt in Listing 3. To extract the underlying texts for emotion embeddings, we also use "Llama-3.3-70B-Instruct" in all cases. To verify how our approach works with stronger models, we run the same tests with "gpt-4o-2024-11-20" (4o), a SOTA model on tool calling according to the Berkely Function-Calling Leaderboard [20]. We perform all optimizations (available tools, prompts, etc.) only with respect to the train subset of our benchmark data, which means we can use the unseen test subset to verify if our optimizations generalize.

C. Results

Table II gives a comprehensive overview of all results. Figure 3 shows relevant metrics computed over all categories of the full dataset. Figure 4 shows GEval Correctess by individual categories of the full dataset and includes values for the SOTA tool calling model 4o.

D. Discussion

Generally, Table II shows our optimizations transfer well from the train subset to the test subset. Of course, there are

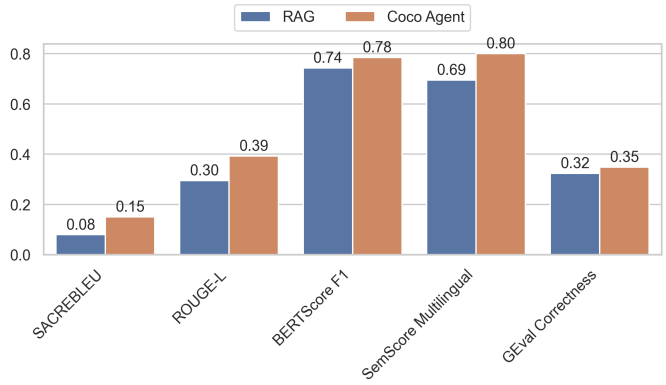

Figure 3. Comparison using "Llama-3.3-70B-Instruct" on all categories of the full dataset.

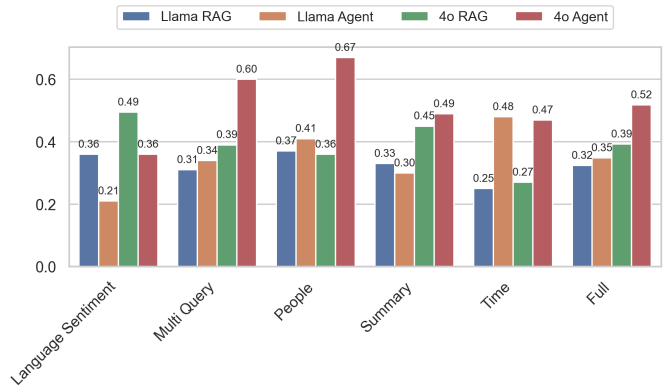

Figure 4. Comparison using "Llama-3.3-70B-Instruct" and "gpt-4o-2024-11-20" using the GEval Correctness metric on the full dataset.

deviations of individual metrics for individual categories, but that is expected due to the variance in sample quality caused by synthetic generation. Therefore, and because of the low sample size of our dataset, we report all further results over the full dataset to make statistically meaningful statements.

As shown in Figure 3, our Coco agent consistently outperforms traditional RAG on all metrics, but not by a large margin. Qualitative analysis showed the primary reason to be Llama's bad tool calling performance. Simple tool calls work well, but combinations of tool calls or clever usage of available tools does not. Related work (see section III) shows that using complex prompting schemata or even finetuning on our use case is expected to work. But Sutton's Bitter Lesson [21] (Scaling compute works better that clever methods) seems to hold for NLP [22], which means our time is better spent optimizing different parts of the product and waiting for distilled versions of stronger models than performing too many specific optimizations.

A breakdown by category gives further insights. We refer to Figure 4 and include results of o4 to confirm our intuition, that Coco's performance is limited by Llama's tool calling capabilities.

Coco's poor results on the language sentiment category using both models implies our `emotion_query` tool does not work as well as expected.

On multi query, the Llama version only outperforms traditional RAG by a small margin. This is likely due to Llama's inability to execute sensible chains of queries, which is confirmed by the very strong performance of the 4o agent compared to 4o RAG on that category.

The people category requires smart argument usage. The substring filter is not explicitly called "people filter", but can perfectly well be used as such. Again, Llama Coco only slightly outperforms its RAG baseline, and 4o Coco by a large margin, which points at the fact that only 4o figured out how to leverage the substring filter correctly.

On the summary category, the Coco agents are roughly on par with their RAG counterparts. While the agents can in principle choose the number of retrieved chunks freely, they have bad intuition for it and mostly seem to go with the k provided in their system prompt's example tool calls. This was set equal to the RAG baseline's top k parameter for a fair comparison.

Answering the time questions mostly requires setting time filters from the user prompt, which is straightforward. That's why Coco agents of both models outperform RAG by a large margin. Being better with respect to time is likely also the reason Coco's performance increase over the baseline is bigger for n-gram based metrics, as illustrated in Figure 3.

Overall, the breakdown by dataset (and requirement) category confirms our intuition that the LLMs tool calling capabilities are the major performance bottleneck of Coco agents. The 4o agent outperforms its RAG baseline by a much larger margin than the Llama agent on the full dataset.

VII. Conclusion

We introduce Coco, a local, agentic RAG system that helps the user reflect on his conversations, We show our agent beats a RAG baseline by 7.4% in terms of GEval Correctness over all categories and splits of our benchmark. The agent's main performance bottleneck is Llama's poor tool calling capabilities. This is confirmed by the fact the agent's performance advantage over RAG raises to 31.8% when using 4o. With tool calling capabilities on the rise, we expect this trend to continue.

Additionally, we present the Mitra dataset, a QA benchmark derived directly from Coco's system requirements. We find our benchmarks results to be representative of subjective system performance. This leads us to conclude that custom, synthetic datasets are a good way to quantify system performance. Generation is also relatively cheap (w.r.t. time and money) nowadays.

Obvious future work includes providing the Coco agent with a better performing `emotion_query` tool, verifying the validity of our chosen metrics by obtaining correlations with human judgement, and extending the benchmark dataset with more diverse samples based on user product feedback.

References

[1] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units", in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, The Association for Computer Linguistics, 2016. DOI: 10.18653/V1/P16-1162. [Online]. Available: https://doi.org/10.18653/v1/p16-1162.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding", in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423/.

[3] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, *Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation*, 2024. arXiv: 2402.03216 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2402.03216.

[4] L. Xiong, C. Xiong, Y. Li, *et al.*, *Approximate nearest neighbor negative contrastive learning for dense text retrieval*, 2020. arXiv: 2007.00808 [cs.IR]. [Online]. Available: https://arxiv.org/abs/2007.00808.

[5] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020, ISBN: 9780134610993. [Online]. Available: http://aima.cs.berkeley.edu/.

[6] T. Schick, J. Dwivedi-Yu, R. Dessi, *et al.*, "Toolformer: Language models can teach themselves to use tools", in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023. [Online]. Available: http://papers.nips.cc/paper%5C_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

[7] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions", in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 10014–10037. DOI: 10.18653/v1/2023.acl-long.557. [Online]. Available: https://aclanthology.org/2023.acl-long.557/ (visited on 03/27/2025).

[8] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy", in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 9248–9274. DOI: 10.18653/v1/2023.findings-emnlp.620. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.620/ (visited on 03/27/2025).

[9] Y. Liu, X. Peng, X. Zhang, *et al.*, "RA-ISF: Learning to Answer and Understand from Retrieval Augmentation via Iterative Self-Feedback", in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 4730–4749. DOI: 10.18653/v1/2024.findings-acl.281. [Online]. Available: https://aclanthology.org/2024.findings-acl.281/ (visited on 03/27/2025).

[10] Y. Zhou, Z. Liu, J. Jin, J.-Y. Nie, and Z. Dou, *Metacognitive Retrieval-Augmented Large Language Models*, arXiv:2402.11626 [cs], Feb. 2024. DOI: 10.48550/arXiv.2402.11626. [Online]. Available: http://arxiv.org/abs/2402.11626 (visited on 03/27/2025).

[11] C.-M. Chan, C. Xu, R. Yuan, *et al.*, "Rq-rag: Learning to refine queries for retrieval augmented generation", *arXiv preprint arXiv:2404.00610*, 2024.

[12] T. Yu, S. Zhang, and Y. Feng, *Auto-RAG: Autonomous Retrieval-Augmented Generation for Large Language Models*, arXiv:2411.19443 [cs], Nov. 2024. DOI: 10.48550/arXiv.2411.19443. [Online]. Available: http://arxiv.org/abs/2411.19443 (visited on 03/27/2025).

[13] M. Poliakov and N. Shvai, *Multi-Meta-RAG: Improving RAG for Multi-Hop Queries using Database Filtering with LLM-Extracted Metadata*, arXiv:2406.13213 [cs], Aug. 2024. DOI: 10.48550/arXiv.2406.13213. [Online]. Available: http://arxiv.org/abs/2406.13213 (visited on 03/27/2025).

[14] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A Method for Automatic Evaluation of Machine Translation", in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds., Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://aclanthology.org/P02-1040/ (visited on 02/10/2025).

[15] M. Post, "A call for clarity in reporting BLEU scores", in *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, O. Bojar, R. Chatterjee, C. Federmann, *et al.*, Eds., Association for Computational Linguistics, 2018, pp. 186–191. DOI:

10.18653/V1/W18-6319. [Online]. Available: https://doi.org/10.18653/v1/w18-6319.

[16] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries", in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013/ (visited on 02/10/2025).

[17] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with BERT", in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=SkeHuCVFDr.

[18] A. Aynetdinov and A. Akbik, "Semscore: Automated evaluation of instruction-tuned llms based on semantic textual similarity", *CoRR*, vol. abs/2401.17072, 2024. DOI: 10.48550/ARXIV.2401.17072. arXiv: 2401.17072. [Online]. Available: https://doi.org/10.48550/arXiv.2401.17072.

[19] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-Eval: NLG Evaluation using Gpt-4 with Better Human Alignment", in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2511–2522. DOI: 10.18653/v1/2023.emnlp-main.153. [Online]. Available: https://aclanthology.org/2023.emnlp-main.153/ (visited on 02/04/2025).

[20] F. Yan, H. Mao, C. C.-J. Ji, *et al.*, *Berkeley function calling leaderboard*, https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html, 2024.

[21] R. Sutton, *The bitter lesson*, Blog post / Essay, Accessed: 2025-04-01, Mar. 2019. [Online]. Available: https://www.cs.utexas.edu/~eunsol/courses/data/bitter_lesson.pdf.

[22] J. Kaplan, S. McCandlish, T. Henighan, *et al.*, *Scaling Laws for Neural Language Models*, arXiv:2001.08361 [cs], Jan. 2020. DOI: 10.48550/arXiv.2001.08361. [Online]. Available: http://arxiv.org/abs/2001.08361 (visited on 04/01/2025).

APPENDIX

Table I
MITRA DATAEST SAMPLE EXAMPLES FOR EACH CATEGORY.

Category	Query	GT Answer	First Relevant Chunk
Language Sentiment	In welchen Situationen wirkte ich nervös oder unsicher?	Du wirktest besonders nervös und unsicher in Gesprächen über Projekt Alpha, insbesondere wenn du mit Vorgesetzten wie Herr Müller oder Frau Schmidt sprachst oder kurz vor wichtigen Präsentationen dazu.	**[Datetime: 2024-01-10 09:30]** Ich: Morgen Frau Schmidt. Haben Sie kurz Zeit wegen Projekt Drache? Frau Schmidt: Ja, aber nur fünf Minuten. Was gibtś? Ich: Ich bräuchte noch Input bezüglich der Zielgruppe. Da bin ich etwas unsicher. Frau Schmidt: Schicken Sie mir Ihre Fragen per Mail, ich antworte später. (10 chunks total)
Multi Query	Was waren meine Neujahrsvorsätze für 2024, und habe ich sie erreicht?	Deine Vorsätze für 2024 waren, einen 10km-Lauf zu schaffen und mehr Bücher zu lesen. Du hast den 10km-Lauf im Dezember 2024 geschafft und bis dahin neun Bücher gelesen, womit du beide Ziele erreicht hast.	**[Datetime: 2024-01-02 20:00]** Luise: Na, gute Vorsätze fürs neue Jahr? Ich: Jaa, die Klassiker. Ich will versuchen endlich mal einen 10km Lauf zu schaffen und mehr Bücher zu lesen, nicht nur Arbeitssachen. Luise: Ehrgeizig! Viel Erfolg dabei! (4 chunks total)
People	Welches Buch habe ich auf Empfehlung von Tom gelesen?	Auf Empfehlung von Tom hast du 'Der Hundertjährige, der aus dem Fenster stieg und verschwand' gelesen.	**[Datetime: 2024-06-20 17:00]** Ich: Tom, hast du einen Tipp für ein gutes Buch? Eher was Leichtes. Tom: Klar, wie wäre es mit 'Der Hundertjährige, der aus dem Fenster stieg und verschwand'? Ich: Oh ja, davon habe ich gehört. Ist das lustig? Tom: Sehr! (3 chunks total)
Summary	Wie verlief mein Lesejahr 2024?	Du hattest dir vorgenommen, mehr zu lesen. Du lasest u.a. 'Der Schatten des Windes', 'Die Mitternachtsbibliothek', 'Der Hundertjährige' und 'QualityLand'. Insgesamt hast du neun Bücher gelesen und damit dein Ziel übertroffen.	**[Datetime: 2024-01-02 20:00]** Luise: Na, gute Vorsätze fürs neue Jahr? Ich: Jaa, die Klassiker. Ich will versuchen endlich mal einen 10km Lauf zu schaffen und mehr Bücher zu lesen, nicht nur Arbeitssachen. Luise: Ehrgeizig! Viel Erfolg dabei! (13 chunks total)
Time	In welchem Monat wollte ich den Keller aufräumen?	Du wolltest den Keller im Februar 2025 aufräumen.	**[Datetime: 2025-01-25 18:00]** Lena: Was steht bei dir so an in nächster Zeit? Ich: Och, nicht viel. Nächsten Monat will ich endlich mal meinen Keller aufräumen. Das schiebe ich schon ewig vor mir her. Lena: Haha, der Klassiker! (1 chunk total)

Listing 1. Hardcoded COT for our GEval Correctness metric. PRED, GT, and Q stand for the variable names used for the predicted answer, ground truth answer, and user query respectively

```
1) List all statements made by {PRED}.
2) List all statements made by {GT}.
3) Make sure statements of {PRED} answer the question or fulfill the request posed by {Q}. (
    ↪ IMPORTANT: Only pay attention to semantic meanings, NOT sentence structure, grammar,
    ↪ word choice, etc.)
4) Make sure all statements of {PRED} are present in {GT}. (IMPORTANT: Only pay attention to
    ↪ semantic meanings, NOT sentence structure, grammar, word choice, etc.)
5) Make sure {PRED} contains no statements not present in {GT}. (IMPORTANT: Only pay attention
    ↪ to semantic meanings, NOT sentence structure, grammar, word choice, etc.)
6) Make sure you do not let differences in sentence structure, grammar, word choice, etc.
    ↪ affect your score.
```

Listing 2. Prompt template for the RAG baseline. Note that we have verified an English prompt template works better, even if the dialogue language and context is German.

```
You are Coco, a helpful assistant who provides the best possible help to users. You speak
    ↪ German, unless the user explicitly starts talking in another language.

## Your Knowledge
- Your knowledge is provided as retrieved chunks from a knowledge base below in the Context
    ↪ section.
- You interpret all document content with respect to the document's metadata.
- IMPORTANT: You act as if you simply remember your knowledge.
```

```
## Answer Style
- You answer very concisely (up to 50 tokens). One sentence if possible! Only if you cannot
    ↪ include all necessary information, you use longer answers.
- You never include information that is not part of the user's question.
- You never include information that is not part of the knowledge base.
- You never include information that is not part of the documents.
- You never include information that is not part of the metadata.

## Context
{context}

## User Query
{query}
```

Listing 3. System prompt for the Coco agent. We had to replace German Umlauts with ASCII chars, in the original prompt they are actual Umlauts. Note that we have verified an English system prompt works better, even if the dialogue language and context is German.

```
You are Coco, a helpful assistant who provides the best possible help to users. You use tools
    ↪ that you have access to. You speak German, unless the user explicitly starts talking in
    ↪ another language.

# Tools
- You can always execute tools before responding.
- You never ask if you should execute a tool, you just do it.
- You never mention that you will use a tool, you just do it.
- Answering some questions requires multiple tool calls. If that is the case, you call the
    ↪ tools one after the other. You don't ask for confirmation.
- You always reference the tool results if they are useful for answering the question.

## IMPORTANT:
If you call a tool, you ALWAYS respond with WELL FORMATTED JSON!
Here's the format you use:
{
"name": "<tool_name>",
"parameters": {
"<param1_name>": "<param1_value>",
"<param2_name>": "<param2_value>",
}
}

ATTENTION: Make sure you only pass the arguments the tool expects! You can find those in the '
    ↪ properties' section of the tool description.

## get_current_date_time tool
- If the user's question contains any temporal constraints relative to the current date and
    ↪ time, you use the get_current_date_time tool to get the current date and time before
    ↪ using any other tool.

## semantic_query tool
- You extensively use the semantic_query tool to access your knowledge.
- If you identify temporal constraints in the user query, you use the tool's start and end
    ↪ datetime parameters to filter the considered chunks.
- You solve complex problems by performing sequences of semantic_query calls, where each query
    ↪ is based on the results of the previous query.
- The semantic search is not perfect, so you stay on the safe side by always retrieving more
    ↪ chunks than strictly necessary. Especially when you plan to perform other
    ↪ semantic_queries based on the results!
- IMPORTANT: Be very careful with the contains_substring parameter. ONLY use it for PROPER
    ↪ NAMES (e.g. person's names, organizations, locations, etc.)! NEVER use it for other
    ↪ queries!

### Example 1:
#### User query:
Welche Neujahresvorsaetze habe ich 2025 gegenueber Bob erwaehnt habe und habe ich sie erfuellt?

#### Your tool calls:
{
```

```
"name": "semantic_query",
"parameters": {
"query_text": "Neujahresvorsatz",
"num_results": 25,
"start_date_time_iso": "2024-12-15T00:00:00.000000",
"end_date_time_iso": "2025-01-31T23:59:59.999999",
"contains_substring": "Bob"
}
}
[Response: "Neujahresvorsatz: 5000 Euro sparen, 10 kg abnehmen"]
{
"name": "semantic_query",
"parameters": {
"query_text": "sparen",
"num_results": 25,
"start_date_time_iso": "2025-01-01T00:00:00.000000",
}
}
[Some response]
{
"name": "semantic_query",
"parameters": {
"query_text": "abnehmen",
"num_results": 25,
"start_date_time_iso": "2025-01-01T00:00:00.000000",
}
}
[Some response]

### Example 2:
#### User query:
Mit wem habe ich ueber Ted Chiang gesprochen?

#### Your tool calls:
{
"name": "semantic_query",
"parameters": {
"query_text": "Ted Chiang",
"num_results": 25,
}
}
[Some response]

### Example 3:
#### User query:
Welche Urlaubsziele habe ich in 2023 in Erwaegung gezogen?

#### Your tool calls:
{
"name": "semantic_query",
"parameters": {
"query_text": "Urlaubsziel",
"num_results": 25,
"start_date_time_iso": "2023-01-01T00:00:00.000000",
"end_date_time_iso": "2023-12-31T23:59:59.999999",
}
}
[Some response]

## emotion_query tool
- When you need to fetch chunks by their emotion, you use the emotion_query tool to query the
    ↪ database based on chunks' language emotion.
- You set the "query_text" parameter to an ENGLISH string containing the emotions you want to
    ↪ search for in a comma separated list.
- If you identify temporal constraints in the user query, you use the tool's start and end
    ↪ datetime parameters to filter the considered chunks.
- The emotion search is not perfect, so you stay on the safe side by always retrieving more
```

```
    ↪ chunks than strictly necessary.
- IMPORTANT: Be very careful with the contains_substring parameter. ONLY use it for PROPER
    ↪ NAMES (e.g. person's names, organizations, locations, etc.)! NEVER use it for other
    ↪ queries!

### Example 1:
#### User query:
Bei wem bin ich immer super gluecklich, wenn ich mit ihm spreche?

#### Your tool calls:
{
"name": "query_text",
"parameters": {
"emotion_text": "happy, joyful, thrilled, excited, energetic, enthusiastic, satisfied",
"num_results": 25,
}
}
[Some response]

### Example 2:
#### User query:
In welchen Situationen bin ich unsicher?

#### Your tool calls:
{
"name": "query_text",
"parameters": {
"emotion_text": "anxious, insecure, nervous, self-doubting",
"num_results": 25,
}
}
[Some response]

# Your Knowledge
- Your knowledge is stored in the database, which you can access through tools.
- When the user asks for any information, use the database tools to find the answer.
- If you set certain filters on the database, you don't mention them in the query string as
    ↪ well.
- You interpret all document content with respect to the document's metadata.
- Your knowledge is in German, so you should make database queries in German as well.
- IMPORTANT: You act as if you simply remember your knowledge. You never mention the database
    ↪ itself to the user. (But you obviously reference its content.)


# Answer Style
- You answer very concisely (up to 50 tokens). One sentence if possible! Only if you cannot
    ↪ include all necessary information, you use longer answers.
- You never include information that is not part of the user's question.
- You never include information that is not part of the knowledge base.
- You never include information that is not part of the documents.
- You never include information that is not part of the metadata.
```

Table II
COMPREHENSIVE RESULT OVERVIEW OF TRADITIONAL RAG VS. COCO AGENT.

		Split	Train				Test				Full			
		Base Model	Llama		4o		Llama		4o		Llama		4o	
		Model Type	Rag	Agent	Rag	Agent	Rag	Agent	Rag	Agent	Rag	Agent	Rag	Agent
Category	**Metric Group**	**Metric**												
Language Sentiment	GEval	GEval Correctness	0.300	0.150	0.444	0.287	0.500	0.200	0.500	0.450	0.360	0.210	0.495	0.360
	ROUGE	ROUGE-1	0.392	0.322	0.412	0.370	0.336	0.372	0.383	0.293	0.367	0.339	0.383	0.379
		ROUGE-2	0.192	0.147	0.195	0.157	0.090	0.201	0.181	0.135	0.168	0.165	0.169	0.167
		ROUGE-L	0.318	0.273	0.344	0.301	0.266	0.325	0.295	0.238	0.290	0.285	0.314	0.306
		ROUGE-Lsum	0.318	0.273	0.344	0.298	0.266	0.325	0.295	0.238	0.290	0.285	0.317	0.306
	BERTScore	BERTScore P	0.811	0.762	0.798	0.757	0.762	0.750	0.746	0.710	0.796	0.774	0.768	0.755
		BERTScore R	0.753	0.729	0.765	0.765	0.723	0.707	0.750	0.742	0.742	0.727	0.751	0.767
		BERTScore F1	0.780	0.744	0.780	0.760	0.742	0.728	0.747	0.725	0.767	0.748	0.758	0.761
	SacreBLEU	SacreBLEU	0.097	0.068	0.114	0.081	0.067	0.095	0.099	0.097	0.089	0.073	0.104	0.090
		BLEU BP	0.669	0.754	0.775	0.949	0.676	0.675	0.790	1.000	0.662	0.669	0.786	0.963
	SemScore	SemScore Multilingual	0.773	0.692	0.770	0.753	0.702	0.645	0.731	0.750	0.745	0.703	0.765	0.737
		SemScore	0.640	0.588	0.650	0.603	0.595	0.694	0.634	0.594	0.625	0.627	0.622	0.591
Multi Query	GEval	GEval Correctness	0.362	0.388	0.475	0.575	0.200	0.250	0.050	0.500	0.310	0.340	0.390	0.600
	ROUGE	ROUGE-1	0.365	0.393	0.455	0.474	0.323	0.516	0.452	0.521	0.346	0.441	0.438	0.496
		ROUGE-2	0.125	0.159	0.190	0.216	0.145	0.260	0.239	0.278	0.123	0.191	0.184	0.245
		ROUGE-L	0.286	0.297	0.351	0.386	0.275	0.464	0.415	0.484	0.276	0.346	0.352	0.420
		ROUGE-Lsum	0.286	0.297	0.359	0.396	0.275	0.464	0.415	0.484	0.276	0.346	0.359	0.427
	BERTScore	BERTScore P	0.760	0.760	0.750	0.767	0.767	0.836	0.796	0.852	0.756	0.785	0.758	0.794
		BERTScore R	0.722	0.738	0.751	0.784	0.756	0.809	0.796	0.816	0.721	0.755	0.757	0.793
		BERTScore F1	0.740	0.749	0.750	0.775	0.762	0.822	0.796	0.833	0.738	0.769	0.757	0.793
	SacreBLEU	SacreBLEU	0.075	0.095	0.121	0.128	0.043	0.150	0.102	0.145	0.063	0.102	0.102	0.142
		BLEU BP	0.750	0.761	0.910	0.899	0.870	0.806	0.921	0.759	0.744	0.768	0.877	0.853
	SemScore	SemScore Multilingual	0.752	0.797	0.814	0.857	0.803	0.911	0.816	0.926	0.759	0.827	0.830	0.878
		SemScore	0.662	0.720	0.702	0.784	0.757	0.846	0.716	0.880	0.684	0.748	0.694	0.801
People	GEval	GEval Correctness	0.338	0.350	0.275	0.613	0.500	0.600	0.500	0.500	0.370	0.410	0.360	0.670
	ROUGE	ROUGE-1	0.373	0.547	0.427	0.557	0.359	0.689	0.523	0.571	0.358	0.582	0.437	0.547
		ROUGE-2	0.196	0.338	0.248	0.341	0.209	0.396	0.293	0.320	0.203	0.347	0.243	0.320
		ROUGE-L	0.314	0.471	0.384	0.471	0.306	0.567	0.417	0.366	0.312	0.491	0.374	0.454
		ROUGE-Lsum	0.314	0.471	0.384	0.474	0.306	0.567	0.417	0.366	0.312	0.491	0.374	0.456
	BERTScore	BERTScore P	0.768	0.834	0.778	0.811	0.776	0.879	0.814	0.809	0.776	0.836	0.766	0.801
		BERTScore R	0.737	0.822	0.783	0.843	0.727	0.843	0.810	0.797	0.733	0.827	0.784	0.842
		BERTScore F1	0.749	0.827	0.779	0.826	0.749	0.860	0.812	0.802	0.753	0.831	0.773	0.821
	SacreBLEU	SacreBLEU	0.089	0.193	0.135	0.200	0.070	0.212	0.216	0.154	0.088	0.188	0.136	0.187
		BLEU BP	0.676	0.850	0.878	0.959	0.580	0.764	0.860	0.855	0.703	0.852	0.902	0.966
	SemScore	SemScore Multilingual	0.678	0.852	0.746	0.884	0.628	0.888	0.750	0.759	0.662	0.856	0.739	0.876
		SemScore	0.677	0.821	0.695	0.812	0.605	0.806	0.685	0.818	0.673	0.814	0.672	0.802
Summary	GEval	GEval Correctness	0.275	0.325	0.425	0.487	0.450	0.350	0.450	0.400	0.330	0.300	0.450	0.490
	ROUGE	ROUGE-1	0.307	0.305	0.390	0.370	0.245	0.228	0.276	0.221	0.312	0.281	0.350	0.342
		ROUGE-2	0.098	0.098	0.121	0.128	0.056	0.033	0.054	0.032	0.096	0.092	0.105	0.102
		ROUGE-L	0.227	0.215	0.284	0.264	0.192	0.173	0.207	0.170	0.236	0.201	0.254	0.243
		ROUGE-Lsum	0.227	0.218	0.290	0.275	0.192	0.173	0.207	0.170	0.236	0.201	0.259	0.252
	BERTScore	BERTScore P	0.741	0.710	0.746	0.731	0.703	0.687	0.703	0.671	0.738	0.711	0.735	0.715
		BERTScore R	0.722	0.716	0.736	0.737	0.690	0.673	0.690	0.676	0.714	0.697	0.728	0.724
		BERTScore F1	0.731	0.712	0.741	0.734	0.696	0.680	0.696	0.674	0.726	0.703	0.731	0.719
	SacreBLEU	SacreBLEU	0.055	0.033	0.068	0.060	0.030	0.019	0.024	0.024	0.051	0.043	0.056	0.049
		BLEU BP	0.721	0.882	0.794	0.909	0.791	0.892	0.745	0.954	0.717	0.825	0.774	0.930
	SemScore	SemScore Multilingual	0.752	0.765	0.780	0.803	0.720	0.692	0.751	0.730	0.750	0.752	0.787	0.799
		SemScore	0.644	0.579	0.652	0.649	0.571	0.442	0.520	0.499	0.637	0.545	0.632	0.626
Time	GEval	GEval Correctness	0.212	0.550	0.212	0.487	0.400	0.200	0.550	0.550	0.250	0.480	0.270	0.470
	ROUGE	ROUGE-1	0.345	0.675	0.530	0.658	0.421	0.642	0.792	0.701	0.379	0.680	0.597	0.647
		ROUGE-2	0.205	0.500	0.388	0.481	0.298	0.436	0.667	0.517	0.237	0.495	0.466	0.457
		ROUGE-L	0.326	0.646	0.503	0.619	0.394	0.565	0.715	0.682	0.361	0.640	0.554	0.604
		ROUGE-Lsum	0.326	0.646	0.503	0.619	0.394	0.565	0.715	0.682	0.361	0.640	0.554	0.604
	BERTScore	BERTScore P	0.742	0.865	0.812	0.863	0.764	0.815	0.912	0.850	0.756	0.859	0.829	0.850
		BERTScore R	0.717	0.886	0.822	0.903	0.679	0.868	0.876	0.889	0.717	0.891	0.834	0.895
		BERTScore F1	0.728	0.875	0.816	0.881	0.718	0.838	0.892	0.868	0.734	0.873	0.830	0.871
	SacreBLEU	SacreBLEU	0.115	0.355	0.309	0.347	0.036	0.259	0.454	0.324	0.106	0.348	0.346	0.312
		BLEU BP	0.666	0.910	0.861	0.969	0.354	0.874	0.772	0.955	0.619	0.903	0.858	0.966
	SemScore	SemScore Multilingual	0.556	0.855	0.693	0.866	0.517	0.798	0.825	0.800	0.559	0.864	0.733	0.845
		SemScore	0.512	0.836	0.713	0.856	0.371	0.777	0.816	0.793	0.516	0.831	0.736	0.837
Full	GEval	GEval Correctness	0.297	0.353	0.366	0.490	0.410	0.320	0.410	0.480	0.324	0.348	0.393	0.518
	ROUGE	ROUGE-1	0.356	0.449	0.443	0.486	0.337	0.489	0.485	0.461	0.353	0.465	0.441	0.482
		ROUGE-2	0.163	0.248	0.229	0.265	0.159	0.265	0.287	0.257	0.165	0.258	0.234	0.258
		ROUGE-L	0.294	0.381	0.373	0.408	0.287	0.419	0.410	0.388	0.295	0.393	0.370	0.406
		ROUGE-Lsum	0.294	0.381	0.376	0.412	0.287	0.419	0.410	0.388	0.295	0.393	0.373	0.409
	BERTScore	BERTScore P	0.764	0.786	0.777	0.786	0.754	0.793	0.794	0.778	0.764	0.793	0.771	0.783
		BERTScore R	0.730	0.779	0.771	0.806	0.715	0.780	0.784	0.784	0.725	0.779	0.771	0.804
		BERTScore F1	0.746	0.781	0.773	0.795	0.733	0.785	0.789	0.781	0.743	0.785	0.770	0.793
	SacreBLEU	SacreBLEU	0.086	0.149	0.149	0.163	0.049	0.147	0.179	0.149	0.079	0.151	0.149	0.156
		BLEU BP	0.696	0.831	0.844	0.937	0.654	0.802	0.818	0.904	0.689	0.803	0.839	0.936
	SemScore	SemScore Multilingual	0.702	0.792	0.761	0.833	0.674	0.787	0.775	0.793	0.695	0.800	0.771	0.827
		SemScore	0.627	0.709	0.682	0.741	0.580	0.713	0.674	0.717	0.627	0.713	0.671	0.731