IEEE/ACM 2024 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN ICCAD 43rd Edition

Customized Retrieval Augmented Generation and Benchmarking for EDA Tool Documentation QA

Yuan Pu[1,2], Zhuolun He[1,2], Tairu Qiu[2], Haoyuan Wu[3], Bei Yu[1]

[1] The Chinese University of Hong Kong
[2] ChatEDA Tech
[3] Shanghai AI Laboratory

Oct. 28, 2024

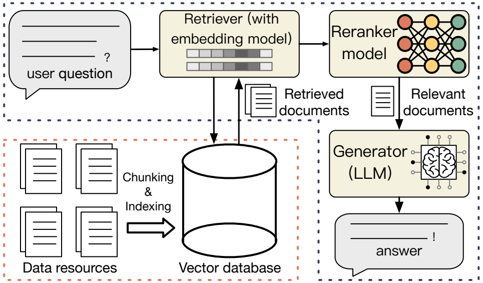
上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

Outline

Introduction

Retrieval Augmented Generation (RAG)

- Key idea: Enhance the output of a generative model by incorporating information retrieved from external knowledge base.
- Modules:
  - Retriever: retrieve documents that may be relevant to the user question.
  - Reranker: Fine-filter the relevant documents.
  - Generator: Answer generation.

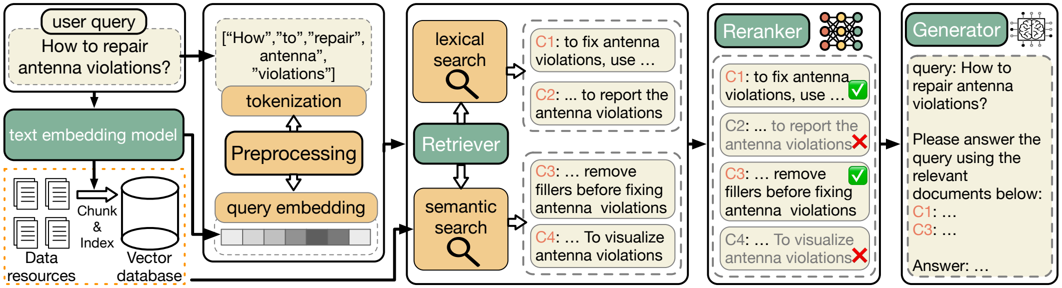


EDA tool Documentation QA

- Pain point: human-involved QA is costly.
- Trend: Automatic question-answering (QA) system for EDA tool documentation.
- Potential solution: RAG.
- Challenges: **Modules** in existing RAG flow lack domain knowledge in EDA:
  - Retriever: low accuracy for relevant document retrieval.
  - Reranker: fail to filter out weakly related document(s).
  - Generator: inadequate domain knowledge and inference capacity for answering EDA-tool question.

Proposed Algorithm

Overall Flow


user query
How to repair antenna violations?
text embedding model
Chunk & Index
Data resources
Vector database
["How","to","repair", antenna", "violations"]
tokenization
Preprocessing
query embedding
lexical search
Retriever
semantic search
C1: to fix antenna violations, use …
C2: ... to report the antenna violations
C3: … remove fillers before fixing antenna violations
C4: … To visualize antenna violations
Reranker
C1: to fix antenna violations, use …
C2: ... to report the antenna violations
C3: … remove fillers before fixing antenna violations
C4: … To visualize antenna violations
Generator
query: How to repair antenna violations?
Please answer the query using the relevant documents below:
C1: …
C3: …
Answer: …

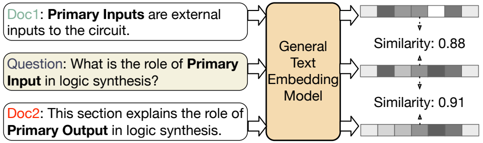
Customization:

- Text embedding model fine-tuning.
- Reranker model fine-tuning.
- Generator pre-train and fine-tuning.

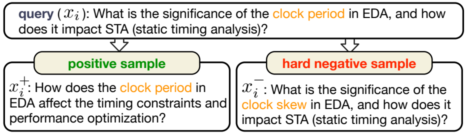
Text Embedding Model Finetuning

- General text embedding model: lacks semantic understanding for EDA-specific terminologies and concepts $\rightarrow$ Low retrieval accuracy.
- Solution: finetune text embedding model by contrastive learning.
- x_i: EDA-related query.
- x_i^+ (pos sample): rephrased sentence of x_i.
- x_i^- (neg sample): Query with another EDA term.
- Loss function:

$$-\log \frac{e^{sim(x_i,x_i^+)/\tau}}{\sum_{j=1}^{M}\left(e^{sim(x_i,x_j^+)/\tau} + e^{sim(x_i,x_j^-)/\tau}\right)}, \quad (1)$$


Doc1: **Primary Inputs** are external inputs to the circuit.
Question: What is the role of **Primary Input** in logic synthesis?
Doc2: This section explains the role of **Primary Output** in logic synthesis.
General Text Embedding Model
Similarity: 0.88
Similarity: 0.91

A fail case of general text embedding model in EDA-specific information retrieval.

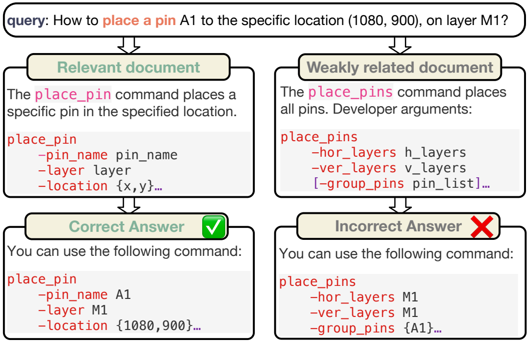

query (x_i): What is the significance of the clock period in EDA, and how does it impact STA (static timing analysis)?
positive sample
x_i^+: How does the clock period in EDA affect the timing constraints and performance optimization?
hard negative sample
x_i^-: What is the significance of the clock skew in EDA, and how does it impact STA (static timing analysis)?

A contrastive data sample used for text embedding model finetuning.

Reranker Model Finetuning

- Weakly related document:
  - Contain the same EDA term as the query.
  - Can not be used to answer the query.
- Distill fine filtering cability of GPT-4: contrastive learning.
- q_i: query; s_i^+: relevant document; s_i^-: weakly related documents filterd by GPT-4.
- Loss:

$$-\log \frac{e^{f(q_i,s_i^+)/\tau}}{e^{f(q_i,s_i^+)/\tau} + \sum_{j=1}^{m} e^{f(q_i,s_{i,j}^-)/\tau}}, \tag{2}$$


query: How to place a pin A1 to the specific location (1080, 900), on layer M1?
Relevant document
Weakly related document
Correct Answer
Incorrect Answer

The `place_pin` command places a specific pin in the specified location.

```
place_pin
    -pin_name pin_name
    -layer layer
    -location {x,y}…
```

The `place_pins` command places all pins. Developer arguments:

```
place_pins
    -hor_layers h_layers
    -ver_layers v_layers
    [-group_pins pin_list]…
```

You can use the following command:

```
place_pin
    -pin_name A1
    -layer M1
    -location {1080,900}…
```

You can use the following command:

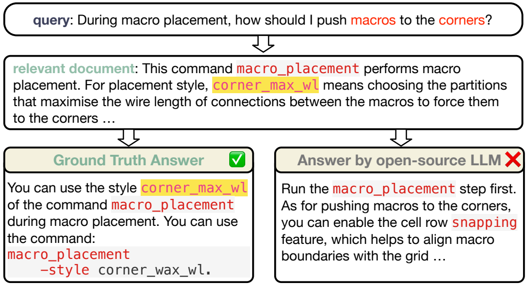
```
place_pins
    -hor_layers M1
    -ver_layers M1
    -group_pins {A1}…
```

A contrastive data sample used for text embedding model finetuning.

Generator Pre-train and Finetuning

- Existing LLMs fail for two reasons:
  - Lack Domain knowledge in EDA.
  - Bad performance in EDA-specific logic inference.
- Two-stage methodology:
  - Domain knowledge pre-train: pretrained two textbooks of EDA.
  - Task-specific fine-tuning: Trained on high-quality EDA-tool QAs.
- Loss function:

$$-\sum_{t=1}^{n} \log P(x_t | x_{<t}; \theta). \qquad (3)$$



query: During macro placement, how should I push macros to the corners?

relevant document: This command `macro_placement` performs macro placement. For placement style, `corner_max_wl` means choosing the partitions that maximise the wire length of connections between the macros to force them to the corners ...

Ground Truth Answer ✓

You can use the style `corner_max_wl` of the command `macro_placement` during macro placement. You can use the command:

```
macro_placement
    -style corner_wax_wl.
```

Answer by open-source LLM ✗

Run the `macro_placement` step first. As for pushing macros to the corners, you can enable the cell row `snapping` feature, which helps to align macro boundaries with the grid ...

A fail case of existing open-source chat LLMs for answering the EDA-tool questions.

Prompt Template for Instruction Tuning

System Prompt:

You are the product consultant of a Electronic Design Automation (EDA) tool **OpenROAD**. Now given the user question and the related reference, you are required to answer the question referring to the provided reference.

During answering the question, you have to follow these instructions:

1. Make your answer as **rigorous** as possible, **do not fabricate** the fact that does not mentioned in the provided reference.
2. Your answer should be strongly related to the provided reference, provide **concrete** solution for the answer, and do not ignore the **precondition** in the query.

Instruction:

Question: During macro placement, how should I push macros to the corners?

Related reference:

Doc1: This command `macro_placement` performs macro placement. For placement style, `corner_max_wl` means choosing the partitions that maximise the wire length of connections between the macros to force them to the corners …

Answer:

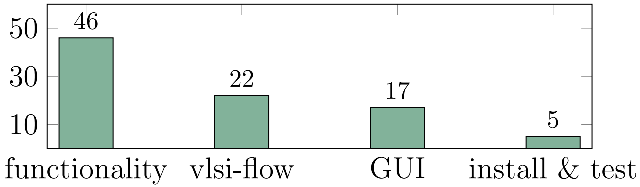
You can use the style `corner_max_wl` of the command `macro_placement` during macro placement. You can use the command:

```
macro_placement -style corner_wax_wl.
```

Experimental Results

Benchmark: ORD-QA

- ORD-QA [1]: 90 high-quality question-document-answer triplets based on the OpenROAD[2] document.
- Three categories of questions:
  - **functionality**: how to introduce randomness to global routing for solution exploration?
  - **VLSI flow**: Assuming a user needs to insert filler cells, analyze clock tree synthesis (CTS), remove the buffers and analyze the net using OpenROAD, how can they go about it?
  - **GUI & Install & Test**: How to create a button in OpenROAD's GUI to load specific LEF files automatically?
- Question number Distribution:



[1]https://github.com/lesliepy99/RAG-EDA

[2]Tutu Ajayi et al. (2019). "Toward an open-source digital flow: First learnings from the openroad project". In: *Proc. DAC*, pp. 1–4.

Experimental Results: Retriever and Rerank

- Metric: recall@k, defined as the proportion of relevant documents that are retrieved among the top k results returned by the retriever/reranker.
- Our embedding model: finetuned on bge-large-en-v1.5.
- Our reranker model: finetuned on bge-reranker-large
- Evaluation:
    - Retriever: top 5, 10, 15, 20.
    - Reranker: First use hybrid retrieve to obtain 20 documents, and test the top-1/2/3/4/5 documents filtered by the reranker model.

Table: Performance of semantic search for text embedding model on ORD-QA.

Model type	recall@5	recall@10	recall@15	recall@20
text-embedding-ada-002	0.447	0.534	0.609	0.634
bge-large-en-v1.5	0.503	0.596	0.634	0.660
Our embedding model	**0.547**	**0.658**	**0.702**	**0.733**

Table: Performance of reranker for weakly-related documents filtering on ORD-QA.

Model type	recall@1	recall@2	recall@3	recall@4	recall@5
RRF	0.248	0.342	0.391	0.435	0.484
bge-reranker-large	**0.360**	0.441	0.484	0.497	0.522
Our reranker	0.335	**0.534**	**0.609**	**0.665**	**0.671**

Experimental Results: Generator

- Performance measurement:
  - Measure the coherence, conciseness and factual consistency of text generation.
  - Metrics: BLEU, ROUGE-L and UniEval.
- RAG-EDA-generator is finetuned on Qwen1.5-14B-chat.
- Performance comparison on ORD-QA with **all ground-truth documented fed**:

Model Type	ORD-QA#functionality			ORD-QA#vlsi-flow			ORD-QA#gui & install & test			ORD-QA#all		
	BLEU	ROUGE-L	UniEval	BLEU	ROUGE-L	UniEval	BLEU	ROUGE-L	UniEval	BLEU	ROUGE-L	UniEval
GPT-4[3]	0.116	0.257	0.724	0.149	0.268	0.759	0.184	0.328	0.798	0.141	0.277	0.751
Qwen1.5-14B-Chat[4]	0.077	0.189	0.604	0.099	0.175	0.513	0.127	0.272	0.739	0.095	0.206	0.615
llama-2-13B-chat[5]	0.099	0.224	0.694	0.110	0.214	0.654	0.125	0.250	0.678	0.108	0.228	0.680
Baichuan2-13B-Chat[6]	0.089	0.244	0.743	0.073	0.260	0.781	0.092	0.281	0.740	0.086	0.257	0.751
RAG-EDA-generator (ours)	**0.150**	**0.319**	**0.788**	**0.188**	**0.326**	**0.798**	**0.224**	**0.374**	**0.809**	**0.177**	**0.334**	**0.795**

[3]Josh Achiam et al. (2023). "Gpt-4 technical report". In: *arXiv preprint*.
[4]Jinze Bai et al. (2023). "Qwen technical report". In: *arXiv preprint*.
[5]Hugo Touvron et al. (2023). "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint*.
[6]Aiyuan Yang et al. (2023). "Baichuan 2: Open large-scale language models". In: *arXiv preprint*.

Experimental Results: RAG-Flow Baselines

RAG-flow baselines:

- Vanilla-RAG[7]: Dense-retrieval + generation.
- RAG-fusion[8]: adopt hybrid-retrival (dense and sparse retrieval).
- HyDE[9]: generate pseudo document for better retrieval.
- llmlingua[10]: prompt compression and key information extraction.
- ITER-RETGEN[11]: Multi-round, generation-guided retrieval.

[7] Patrick Lewis et al. (2020). "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Proc. NIPS* 33, pp. 9459–9474.

[8] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher (2009). "Reciprocal rank fusion outperforms condorcet and individual rank learning methods". In: *Proc. SIGIR*, pp. 758–759.

[9] Luyu Gao et al. (2022). "Precise zero-shot dense retrieval without relevance labels". In: *arXiv preprint*.

[10] Huiqiang Jiang et al. (2023). "LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models". In: *Proc. EMNLP*.

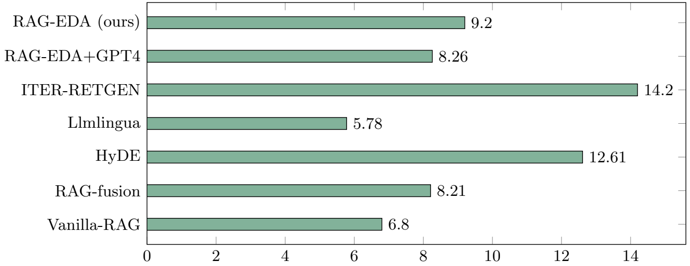
[11] Zhihong Shao et al. (2023). "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy". In: *arXiv preprint*.

Experimental Results: RAG-flow Results

- Test on RAG-flows, documents.
- For baselines, retrieval is by OpenAI text-embeddung-ada-002 model, and generation is by GPT-4.

RAG Flow	ORD-QA#functionality			ORD-QA#vlsi-flow			ORD-QA#gui & install & test			ORD-QA#all		
	BLEU	ROUGE-L	UniEval	BLEU	ROUGE-L	UniEval	BLEU	ROUGE-L	UniEval	BLEU	ROUGE-L	UniEval
Vanilla-RAG	0.088	0.204	0.635	0.089	0.206	0.669	0.139	0.258	0.726	0.101	0.217	0.665
RAG-fusion	0.080	0.193	0.624	0.088	0.202	0.681	0.150	0.274	0.698	0.099	0.215	0.656
HyDE	0.080	0.188	0.603	0.075	0.185	0.658	0.130	0.255	0.712	0.091	0.204	0.643
llmlingua	0.056	0.179	0.565	0.048	0.162	0.586	0.091	0.221	0.658	0.062	0.185	0.593
ITER-RETGEN	0.098	0.208	0.652	0.089	0.202	0.632	0.136	0.251	0.700	0.105	0.217	0.659
RAG-EDA+GPT-4	0.101	0.230	0.681	0.116	0.234	0.690	0.178	0.299	0.742	0.123	0.248	0.698
RAG-EDA (ours)	**0.119**	**0.281**	**0.699**	**0.147**	**0.269**	**0.746**	**0.166**	**0.302**	**0.776**	**0.137**	**0.283**	**0.729**

- Runtime Analysis:



THANK YOU!