

# 高级RAG

## RAG-Fusion原理和代码介绍



可乐i\_Klay

# Forget RAG, the Future is RAG-Fusion

The Next Frontier of Search: Retrieval Augmented Generation meets Reciprocal Rank Fusion and Generated Queries



Adrian H. Raudaschl · [Follow](#)

Published in Towards Data Science · 10 min read · Oct 6

2.5K Q 24

W+ ⏪ ⏹ ...



The Wonderful World of RAG Fusion. Illustration by author.

# RAG-Fusion

RAG

+

Generated Queries

+

Reciprocal Rank Fusion

[Article](#)



可乐i\_Klay

# 为什么是RAG-Fusion

## RAG的局限性

- 词法和向量搜索
- RAG 并不能完全帮助用户问出较全面的问题
- 过度简化搜索

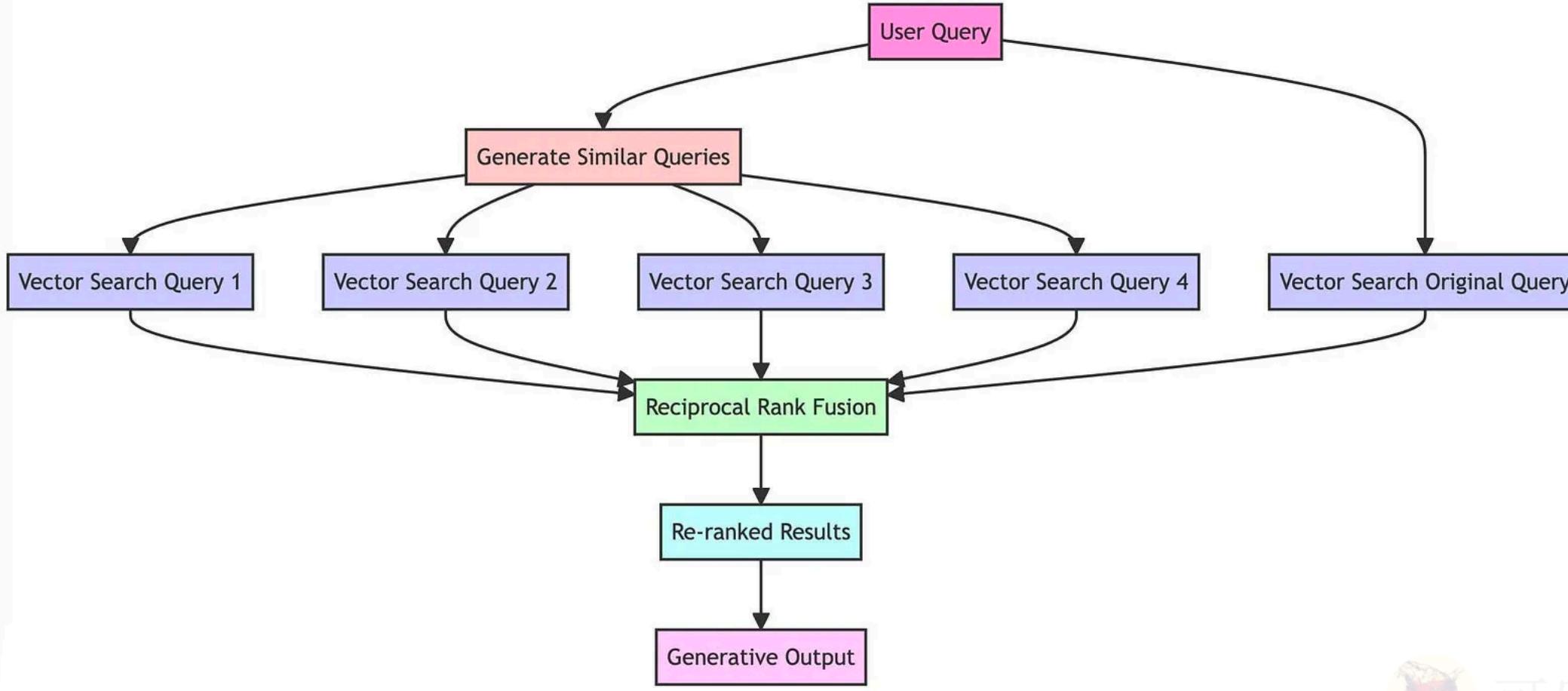
## RAG-Fusion

- 生成多个用户查询并对结果重新排序
- 利用倒数排名融合(RRF)和自定义向量得分加权来获得全面、准确的结果



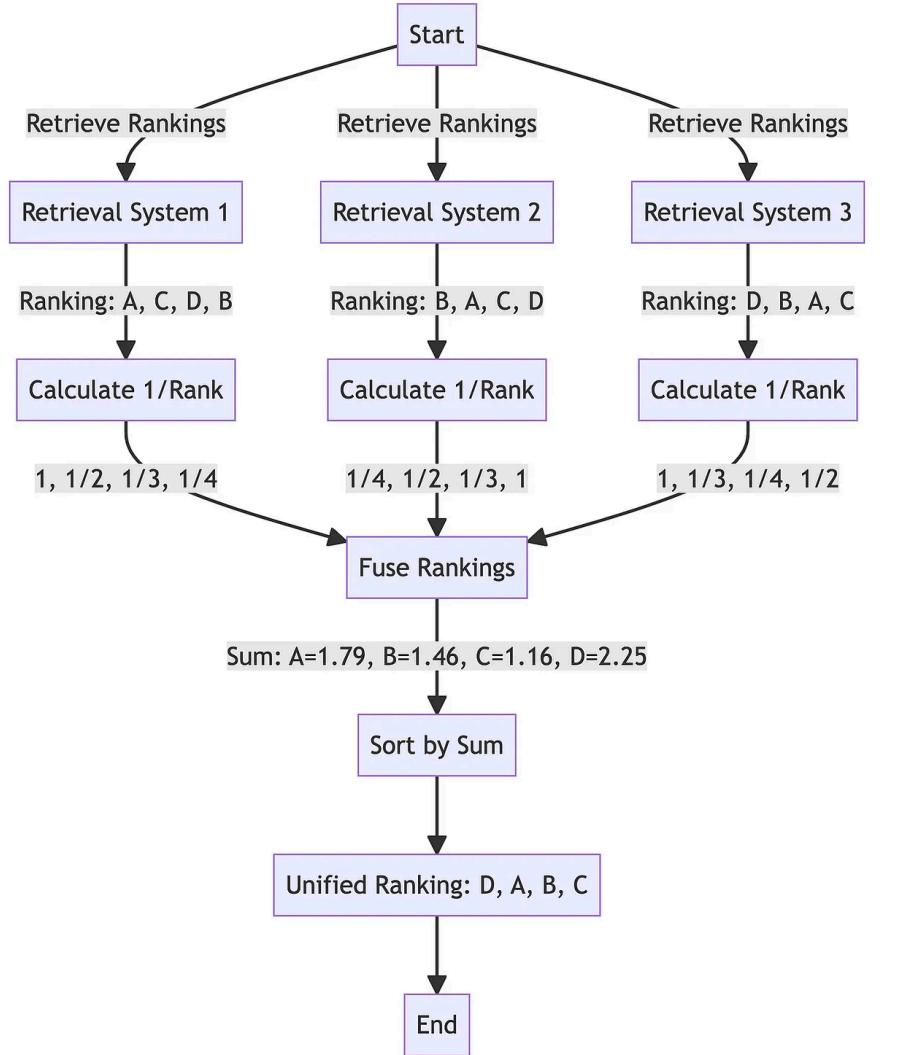
可乐i\_Klay

# RAG -Fusion原理



# 倒数排序融合 Reciprocal Rank Fusion (RRF)

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)},$$



## ChatGPT

Certainly! Let's consider a scenario where we have three systems (System A, System B, and System C) providing rankings for a set of items. For simplicity, let's focus on three items (Item 1, Item 2, and Item 3). The rankings produced by each system are as follows:

### System A:

- Item 1: Rank 2
- Item 2: Rank 1
- Item 3: Rank 3

### System B:

- Item 1: Rank 1
- Item 2: Rank 3
- Item 3: Rank 2

### System C:

- Item 1: Rank 3
- Item 2: Rank 2
- Item 3: Rank 1

Now, let's calculate the reciprocal ranks for each item using the formula mentioned earlier:

$$RR_i = \frac{1}{\text{rank}_A(i)} + \frac{1}{\text{rank}_B(i)} + \frac{1}{\text{rank}_C(i)}$$

### Reciprocal Ranks:

- For Item 1:

$$RR_1 = \frac{1}{2} + \frac{1}{1} + \frac{1}{3} = \frac{13}{6}$$

- For Item 2:

$$RR_2 = \frac{1}{1} + \frac{1}{3} + \frac{1}{2} = \frac{11}{6}$$

- For Item 3:

$$RR_3 = \frac{1}{3} + \frac{1}{2} + \frac{1}{1} = \frac{11}{6}$$



可乐\_i\_Klay

Now, you can use these reciprocal ranks to create a final fused ranking. One common approach is to rank items based on their combined reciprocal ranks. In this case, Item 1 has the highest combined reciprocal rank, so it would be ranked first in the fused ranking.

# RRF代码

```
# Reciprocal Rank Fusion algorithm
def reciprocal_rank_fusion(search_results_dict, k=60):
    fused_scores = {}
    print("Initial individual search result ranks:")
    for query, doc_scores in search_results_dict.items():
        print(f"For query '{query}': {doc_scores}")

    for query, doc_scores in search_results_dict.items():
        for rank, (doc, score) in enumerate(sorted(doc_scores.items(), key=lambda x: x[1], reverse=True)):
            if doc not in fused_scores:
                fused_scores[doc] = 0
            previous_score = fused_scores[doc]
            fused_scores[doc] += 1 / (rank + k)
            print(f"Updating score for {doc} from {previous_score} to {fused_scores[doc]} based on rank {rank} in query '{query}'")

    reranked_results = {doc: score for doc, score in sorted(fused_scores.items(), key=lambda x: x[1], reverse=True)}
    print("Final reranked results:", reranked_results)
    return reranked_results
```



可乐i\_Klay

# Colab代码讲解

Colab code: [RAG-Fusion-OpenAI.ipynb](#)

RAG-Fusion-OpenAI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

- RAG Fusion (OPENAI API KEY Only)

This notebook is to demonstrate an custom example of RAG-Fusion, which aims to bridge the gap between traditional search paradigms and the multifaceted dimensions of human queries, using OPENAI API KEY. Here is the orginal article: [Forget RAG, the Future is RAG-Fusion](#) as well as the [original github repo](#).

Why re-implement this notebook?

The [langchain cookbook](#) and [Sam Witteveen](#) also re-implemented this repo, however, they used different APIs for both LLM, VectorStore and Embeddings. In order to make the life easier, this notebook only needs the OPENAI API Key instead of Pinecone API KEY or Google API KEY.

- Mechanics of RAG-Fusion

```
graph TD; UserQuery[User Query] --> GenSim[Generate Similar Queries]; GenSim --> V1[Vector Search Query 1]; GenSim --> V2[Vector Search Query 2]; GenSim --> V3[Vector Search Query 3]; GenSim --> V4[Vector Search Query 4]; GenSim --> VOrg[Vector Search Original Query]; V1 --> RRF[Reciprocal Rank Fusion]; V2 --> RRF; V3 --> RRF; V4 --> RRF; VOrg --> RRF; RRF --> RRResults[Re-ranked Results]; RRResults --> GO[Generative Output]
```

可乐\_i\_Klay