# Ensemble Retriever

The `EnsembleRetriever` takes a list of retrievers as input and ensemble the results of their get_relevant_documents() methods and rerank the results based on the Reciprocal Rank Fusion algorithm.

By leveraging the strengths of different algorithms, the `EnsembleRetriever` can achieve better performance than any single algorithm.

The most common pattern is to combine a sparse retriever(like BM25) with a dense retriever(like Embedding similarity), because their strengths are complementary. It is also known as "hybrid search".The sparse retriever is good at finding relevant documents based on keywords, while the dense retriever is good at finding relevant documents based on semantic similarity.

```
from langchain.retrievers import BM25Retriever, EnsembleRetriever
from langchain.vectorstores import FAISS
```

**API Reference:**

- BM25Retriever from `langchain.retrievers`
- EnsembleRetriever from `langchain.retrievers`
- FAISS from `langchain.vectorstores`

```
doc_list = [
    "I like apples",
    "I like oranges",
    "Apples and oranges are fruits",
]

# initialize the bm25 retriever and faiss retriever
bm25_retriever = BM25Retriever.from_texts(doc_list)
bm25_retriever.k = 2

embedding = OpenAIEmbeddings()
faiss_vectorstore = FAISS.from_texts(doc_list, embedding)
faiss_retriever = faiss_vectorstore.as_retriever(search_kwargs={"k":
2})
```

```
# initialize the ensemble retriever
ensemble_retriever = EnsembleRetriever(retrievers=[bm25_retriever,
faiss_retriever], weights=[0.5, 0.5])
```

```
docs = ensemble_retriever.get_relevant_documents("apples")
docs
```

```
    [Document(page_content='I like apples', metadata={}),
     Document(page_content='Apples and oranges are fruits', metadata=
{})]
```