



Architectural Components

Let's look at the architectural components of the search ranking system and their role in answering the searcher's query.

We'll cover the following

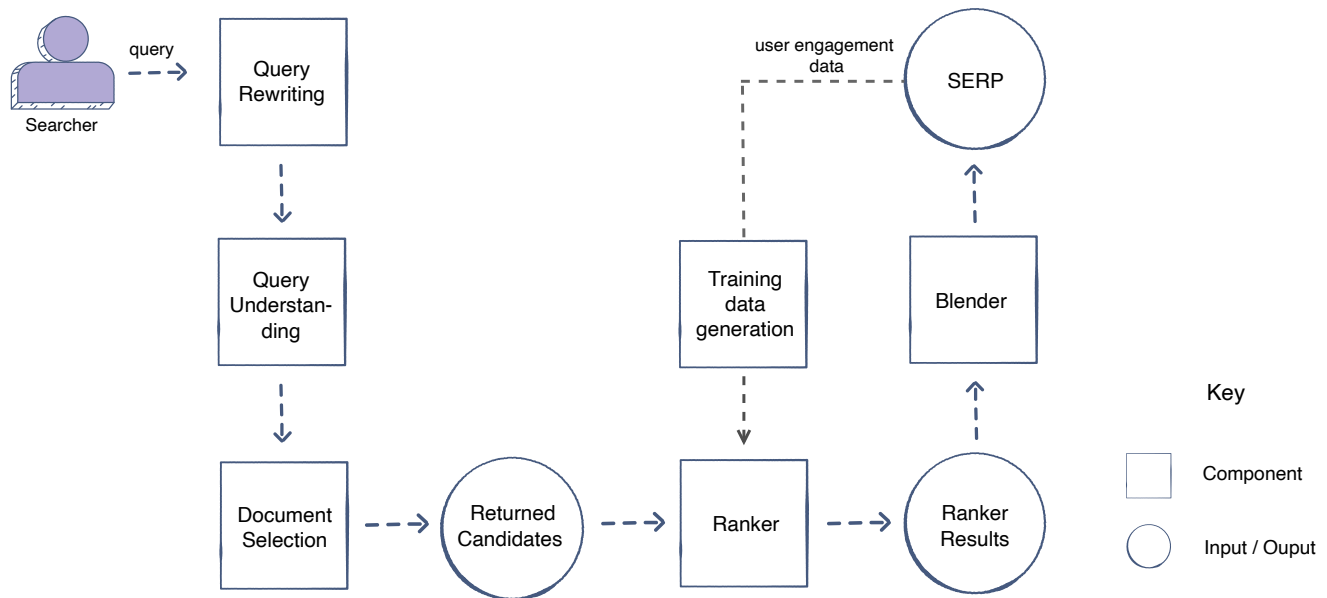


- Architecture
- Query rewriting
 - Spell checker
 - Query expansion
- Query understanding
- Document selection
- Ranker
- Blender
- Training data generation
- Layered model approach

Architecture#

Let's have a look at the architectural components that are integral in creating the search engine.





Architectural components for the search engine

Let's briefly look at each component's role in answering the searcher's query.

Query rewriting#

Queries are often poorly worded and far from describing the searcher's actual information needs. Hence, we use *query rewriting* to increase recall, i.e., to retrieve a larger set of relevant results. Query rewriting has multiple components which are mentioned below.

Spell checker#

Spell checking queries is an integral part of the search experience and is assumed to be a necessary feature of modern search. Spell checking allows you to fix basic spelling mistakes like "itlian restaurat" to "italian restaurant".

Query expansion#


Query expansion improves search result retrieval by adding terms to the

user's query. Essentially, these additional terms minimize the



between the searcher's query and available documents.

Hence, after correcting the spelling mistakes, we would want to expand terms, e.g., for the query "italian restaurant", we should expand "restaurant" to food or recipe to look at all potential candidates (i.e., web pages) for this query.

 The reverse, i.e., *query relaxation*, serves the same purpose. For example, a search for "good italian restaurant" can be relaxed to "italian restaurant".

Query understanding#

This stage includes figuring out the main intent behind the query, e.g., the query "gas stations" most likely has a local intent (an interest in nearby places) and the query "earthquake" may have a newsy intent. Later on, this intent will help in selecting and ranking the best documents for the query. You won't need more details about this for our current scope.

 Newsy intent means the searcher is looking for recent information

Document selection#

The web has billions of documents. Therefore, our first step in document selection is to find a fairly large set of documents that seems relevant to the searcher's query. For example, some common queries like "sports", can match millions of web pages. Document selection's role will be to reduce this set from those millions of documents to a smaller subset of the most relevant



documents.



Document selection is more focused on recall. It uses a simpler technique to sift through billions of documents on the web and retrieve documents that have the potential of being relevant.

Ranking these selected documents in the right order isn't important at this point. We let the *ranking component* worry about finding out “exactly” how relevant (precision) each selected document is and in what order they should be displayed on the SERP.



Since the *ranking component* receives only the documents that have gone through the “initial screening” its workload is greatly reduced. This allows us to use more complex ML modeling options (that have great precision) for the ranking component, without affecting the performance and capacity requirements of the system.

Ranker#

The ranker will actively utilize machine learning to find the best order of documents (this is also called *learning to rank*).

If the number of documents from the document selection stage is significantly large (more than 10k) and the amount of incoming traffic is also huge (more than 10k QPS or queries per second), you would want to have multiple stages of ranking with varying degrees of complexity and model sizes for the ML models. Multiple stages in ranking can allow you to only utilize complex models at the very last stage where ranking order is most important. This keeps computation cost in check for a large scale search system.



For example, one configuration can be that your document selection returns



one, you can use fast (nanoseconds) linear ML models to rank them. In stage two, you can utilise computationally expensive models (like deep learning models) to find the most optimized order of top 500 documents given by stage one.



When choosing an algorithm, remember to consider the model execution time. Cost vs benefit tradeoff is always an important consideration in large scale ML systems.

Blender#

Blender gives relevant results from various search verticals, like, images, videos, news, local results, and blog posts.

For the “italian restaurant” search, we may get a blend of websites, local results, and images. The fundamental motivation behind blending is to satisfy the searcher and to engage them by making the results more relevant.

Another important aspect to consider is the diversity of results, e.g., you might not want to show all results from the same source(website).

The blender finally outputs a *search engine result page* (SERP) in response to the searcher’s query.

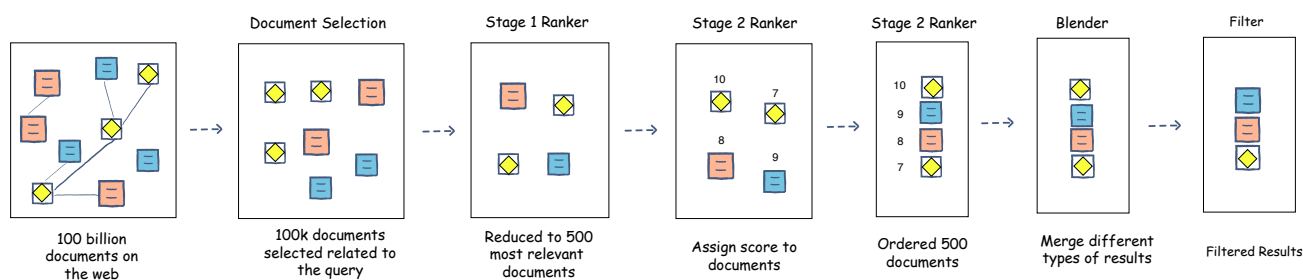
Training data generation#

This component displays the cyclic manner of using machine learning to make a search engine ranking system. It takes online user engagement data from the SERP displayed in response to queries and generates positive and negative training examples. The training data generated is then fed to the



Layered model approach#

Above, we briefly discussed the multi-layered funnel of going from a very large set of documents selected for the query to the topmost relevant one. Let's go over how this configuration might look like for a large scale search system in a bit more detail below.



The layered model approach

These steps translate into a layered model approach that allows you to select the appropriate ML algorithm at each stage, which is thought through the perspective of scalability as well.

The above example configuration assumes that you are first selecting one-hundred thousand documents for the searcher's query from the index, then using two-stage ranking, with the first one reducing from one-hundred thousand to five-hundred documents and the second stage is then ranking these five-hundred documents. The blender can then blend results from different search verticals, and the filter will further screen irrelevant or offensive results to get good user engagement. This is just an example configuration, and it's important to point out that the number of stages and documents ranked at each stage should be selected based on capacity requirements as well as experimentation to see the impact on relevance based on documents scored at each layer.



We will zoom into the different segments of this diagram in the upcoming

lessons.



 **Back**

Metrics

Next 

Document Selection

☒ Mark as Completed

 Report an Issue

