



# Feature Engineering

Let's engineer meaningful features to train the search ranking model.

We'll cover the following



- Features for ML model
  - Searcher-specific features
  - Query-specific features
  - Document-specific features
  - Context-specific features
  - Searcher-document features
  - Query-document features

An important aspect of the feature generation process is to first think about the main actors that will play a key role in our feature engineering process.

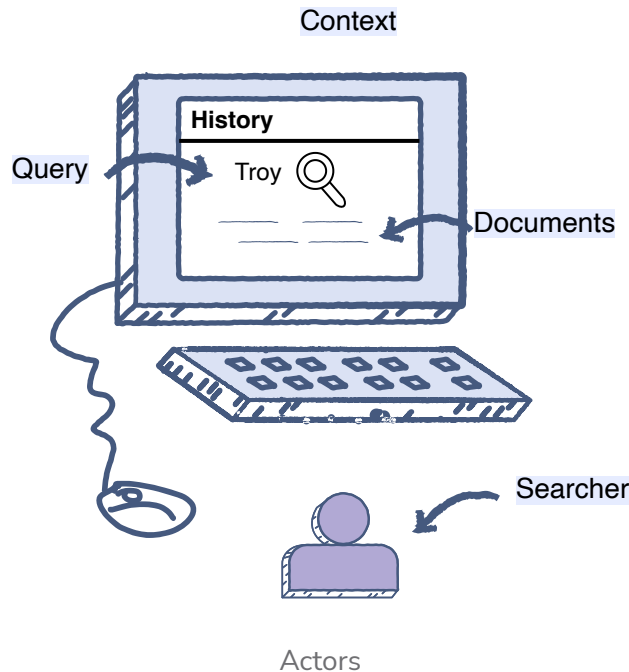


The terms “features” and “signals” are generally used interchangeably as we will also do so.

The **four such actors for search** are:

1. Searcher
2. Query
3. Document
4. Context






In the above figure, the context for a search query is browser history. However, it is a lot more than just search history. It can also include the searcher's age, gender, location, and previous queries and the time of day.

Let's go over the characteristics of these actors and their interactions to generate meaningful features/signals for your machine learning model.

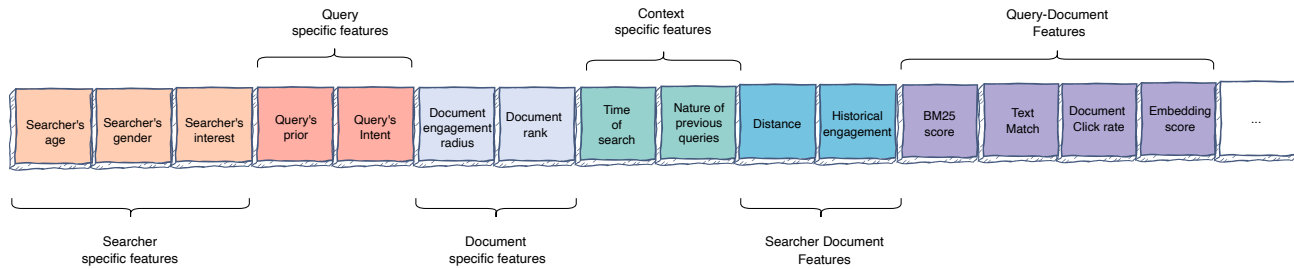
This is essentially the process of feature engineering.

 The knowledge of feature engineering is highly significant from an interview perspective.

## Features for ML model#

We can generate a lot of features for the search ranking problem based on the actors identified above. A subset of these features is shown below.



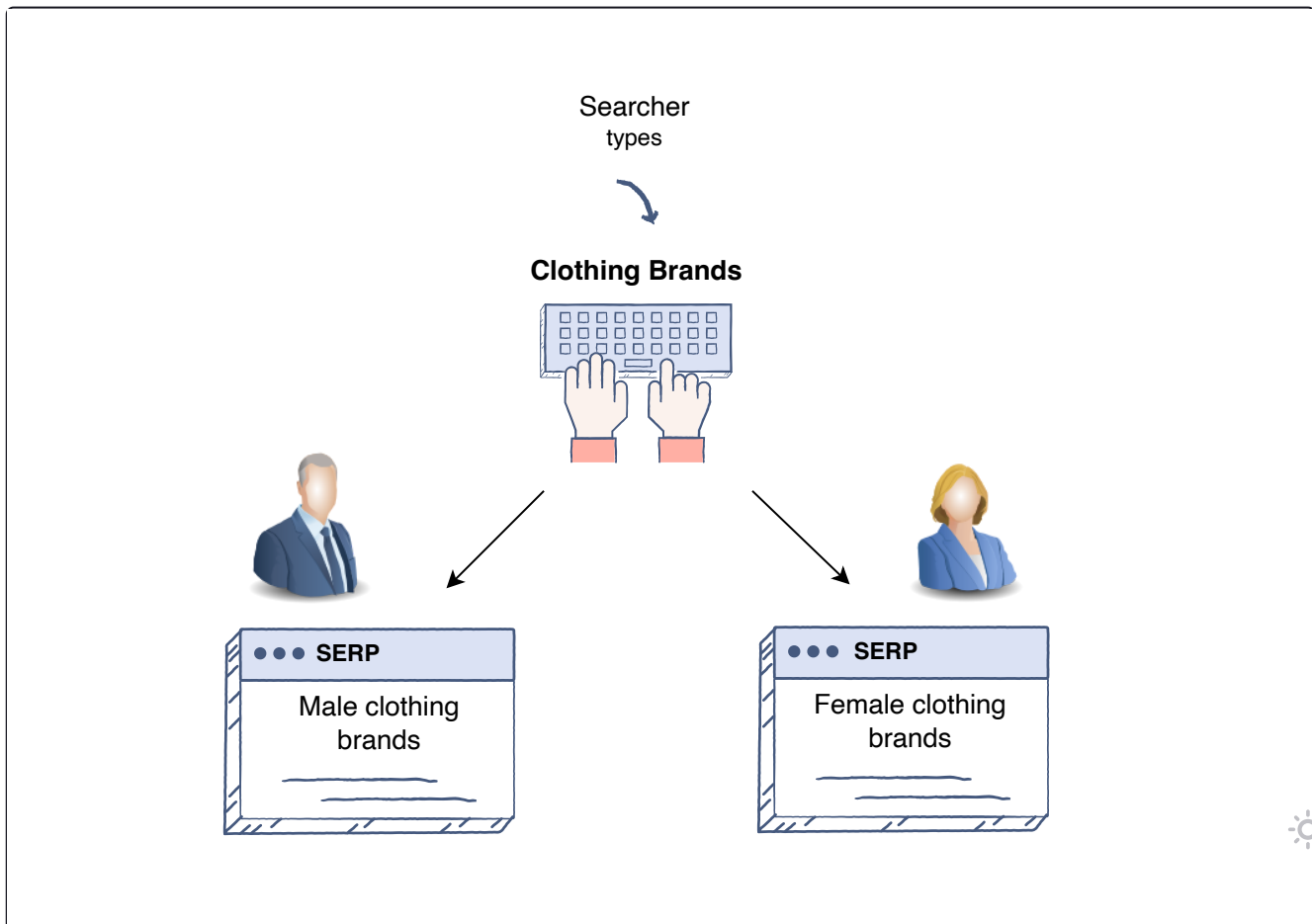


Features in the training data row

Let's discuss these features one by one.

## Searcher-specific features#

Assuming that the searcher is logged in, you can tailor the results according to their age, gender and interests by using this information as features for your model.



Gender-based results



1 of 2



## Query-specific features#

Let's explore some query-related aspects that can also be useful as features.

### Query historical engagement

For *relatively popular queries*, historical engagement can be very important. You can use query's *prior* engagement as a feature. For example, let's say the searcher queries "earthquake". We know from historical data that this query results in engagement with "news component", i.e. most people who searched "earthquake", were looking for news regarding a recent earthquake. Therefore, you should consider this factor while ranking the documents for the query.

### Query intent

The "query intent" feature enables the model to identify the kind of information a searcher is looking for when they type their query. The model uses this feature to assign a higher rank to the documents that match the query's intent. For instance, if a person queries "pizza places", the intent here is *local*. Therefore, the model will give high rank to the pizza places that are located near the searcher.



A few examples of query intent are news, local, commerce, etc.

We can get query intent from the [query understanding component](#).



# Document-specific features#



A lot of features can be engineered with respect to the document's characteristics.

## Page rank

The rank of a document can serve as a feature. To estimate the relevance of the document under consideration, we can look at the number and quality of the documents that link to it.

## Document engagement radius

The *document engagement radius* can be another important feature. A document on a coffee shop in Seattle would be more relevant to people living within a ten-mile radius of the shop. However, a document on the Eiffel Tower might interest people all around the world. Hence, in case our query has a local intent, we will choose the document with the *local scope of appeal* rather than that with a *global scope of appeal*.

# Context-specific features#

We can extract features from the context of the search.

## Time of search

A searcher has queried for restaurants. In this case, a contextual feature can be the *time of the day*. This will allow the model to display restaurants that are open at that hour.


## Recent events

The searcher may appreciate any *recent events related to the query*. For example, upon querying “Vancouver”, the results included:




Vancouver


Top stories



Property values down for most homeowners in Metro Vancouver  
Vancouver Sun · 5 hours ago



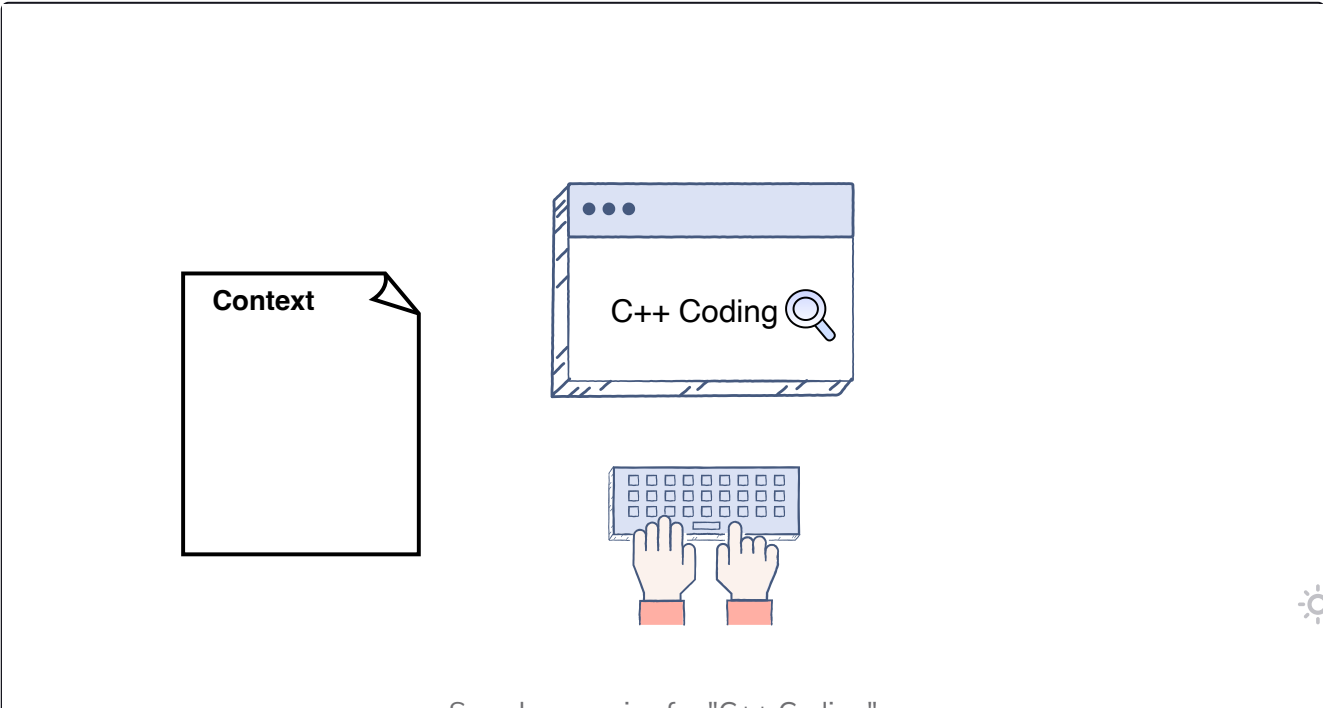
Home values tumble by up to 15% across Metro Vancouver  
CBC.ca · 11 hours ago



Upgrading Skyride a top priority, says Grouse Mountain's new Vancouver-based owners  
CBC.ca · 5 hours ago

➔ [More for vancouver](#)

To provide relevant results for a query, looking at the *nature of the previous queries* can also help. Take a look at the example below.





Now, let's look at the actors jointly to create more features:

## Searcher-document features#

We can also extract features by considering both the searcher and the document.

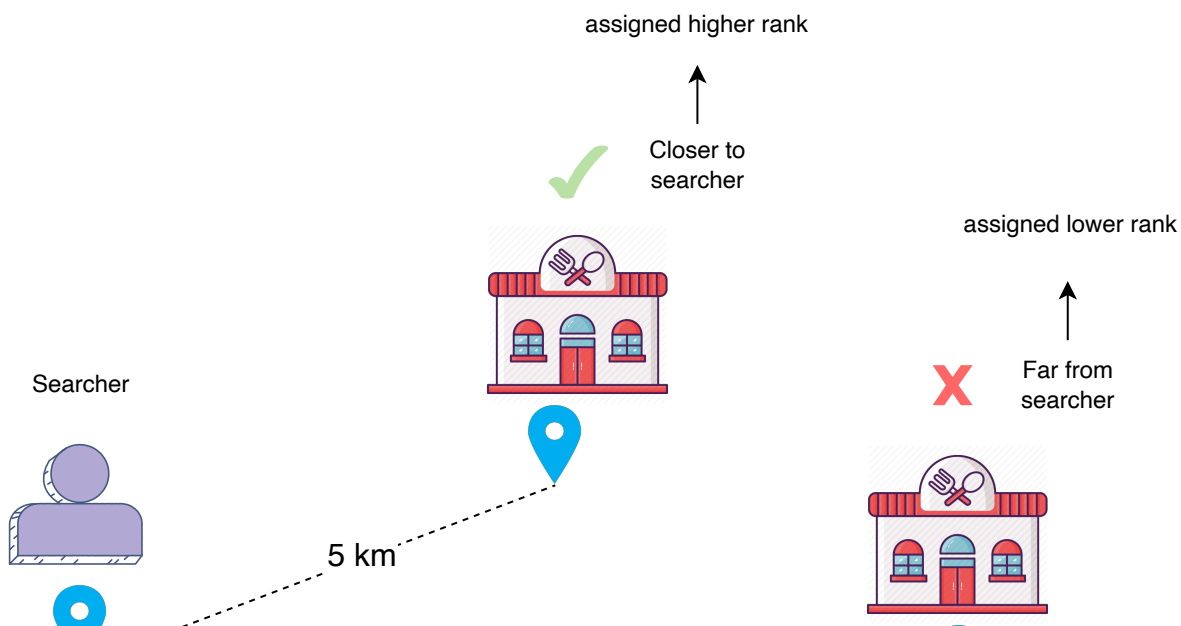
### Distance

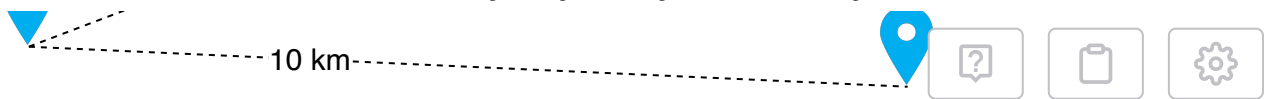
For queries inquiring about nearby locations, we can use the distance between the searcher and the matching locations as a feature to measure the relevance of the documents.

Consider the case where a person has searched for restaurants in their vicinity. Documents regarding nearby restaurants will be selected for ranking. The ranking model can then rank the documents based on the



document.





Feature: the distance between searcher's location and the restaurant's location

## Historical engagement

Another interesting feature could be the searcher's historical engagement with the *result type* of the document. For instance, if a person has engaged with video documents more in the past, it indicates that video documents are generally more relevant for that person.

Historical engagement with a particular website or document can also be an important signal as the user might be trying to “re-find” the document.

## Query-document features#

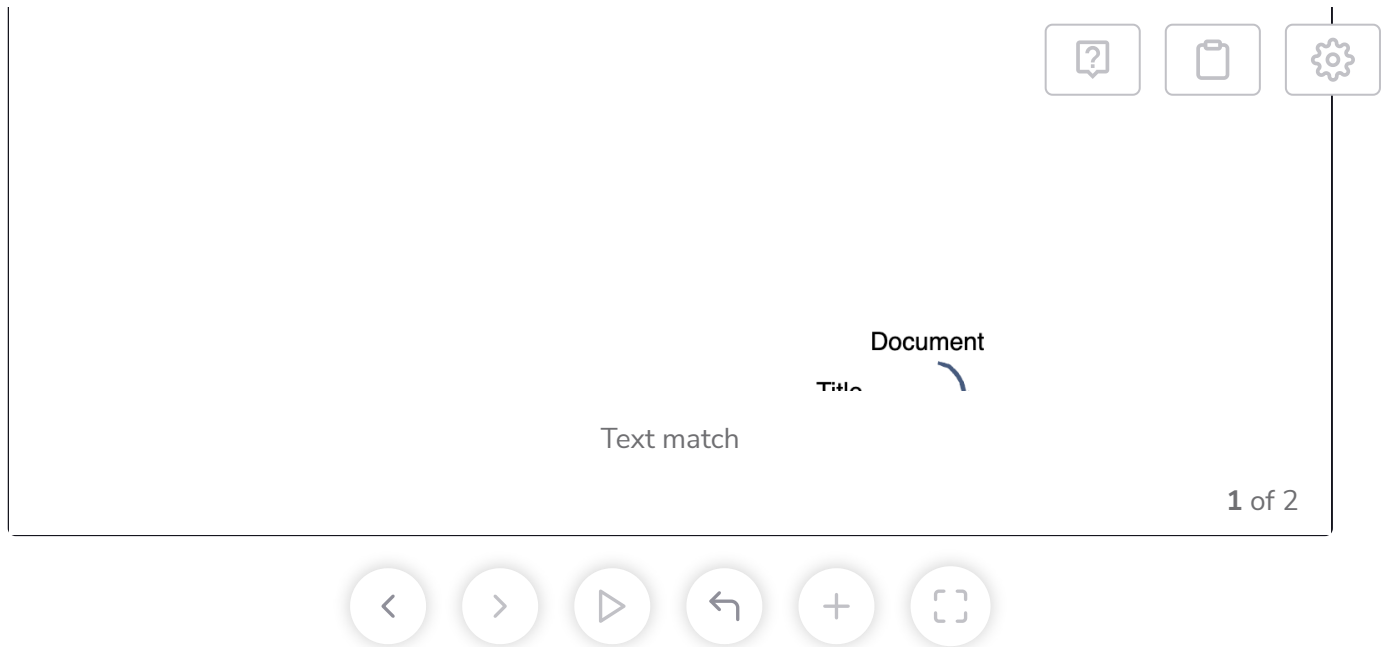
Given the query and the document, we can generate tons of signals.

### Text Match

One feature can be the text match. Text match can not only be in the *title* of the document, but it can also be in the *metadata* or *content* of a document. Look at the following example. It contains a text match between the query and document title and another between the query and document content. These text matches can be used as a feature.







## Unigram or bigram

We can also look at data for each unigram and bigram for text match between the query and document. For instance, the query: “Seattle tourism guide” will result in three unigrams:

1. Seattle
2. tourism
3. guide

These unigrams may match different parts of the document, e.g., “*Seattle*” may match the document title, while “*tourism*” may match the document’s content. Similarly, we can check the match for the bigram and the full trigram, as well. All of these text matches can result in multiple text-based features used by the model.

The **TF-IDF** match score can also be an important relevance signal for the model. It is a similarity score based on a text match between the query terms and the document. TF (term frequency) incorporates the importance of each term for the document and IDF (inverse document frequency) tells us about how much information a particular term provides.



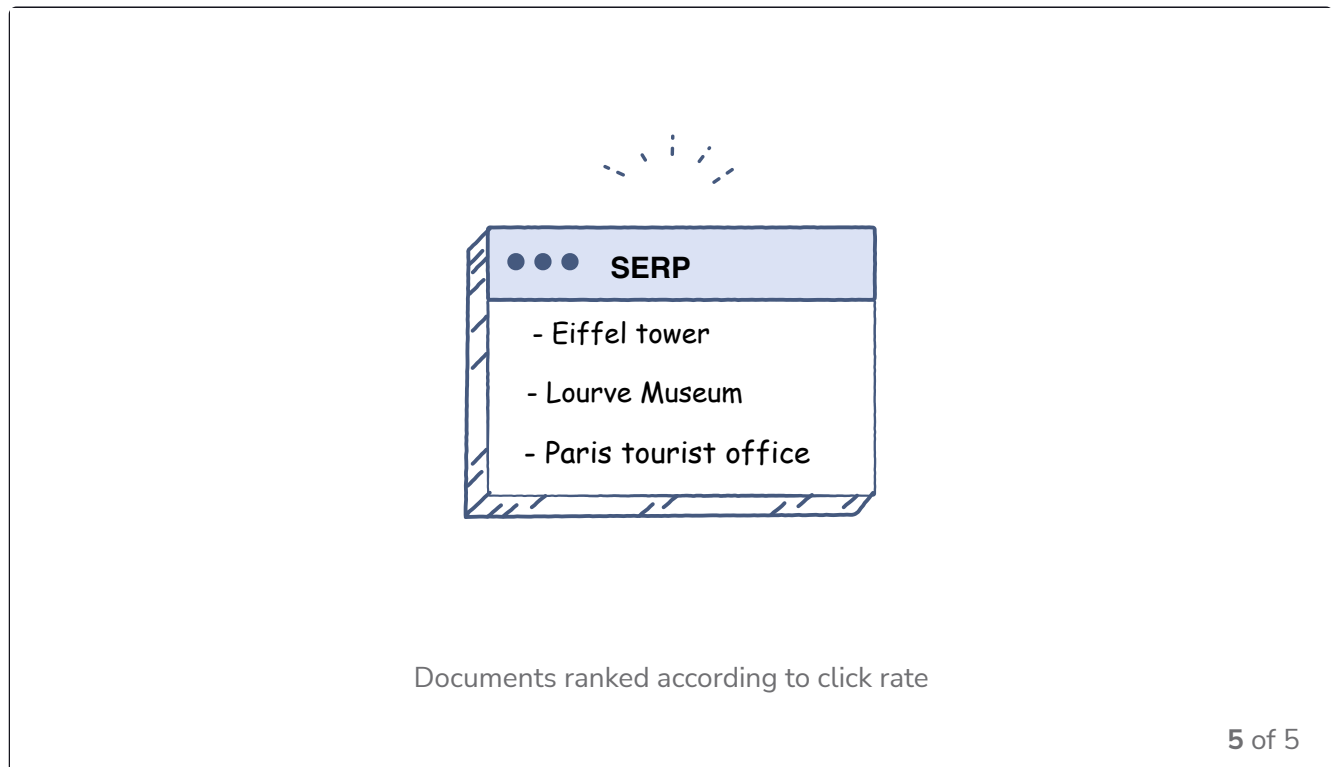
## Query-document historical engagement



The prior engagement data can be a beneficial feature for determining the best ranking of the search results.

- **Click rate**

We want to see users' historical engagement with documents shown in response to a particular query. The click rates for the documents can help in the ranking process. For example, we might observe across people's queries on "Paris tourism" that the click rate for the "Eiffel tower website" is the highest. So, the model will develop the understanding that whenever someone queries "Paris tourism", the document/website on Eiffel tower is the most engaged with. It can then use this information in the ranking of documents.



## Embeddings



We can use embedding models to represent the query and documents in the form of vectors. These vectors can provide significant insight into the relationship between the query and the document. Let's see how.

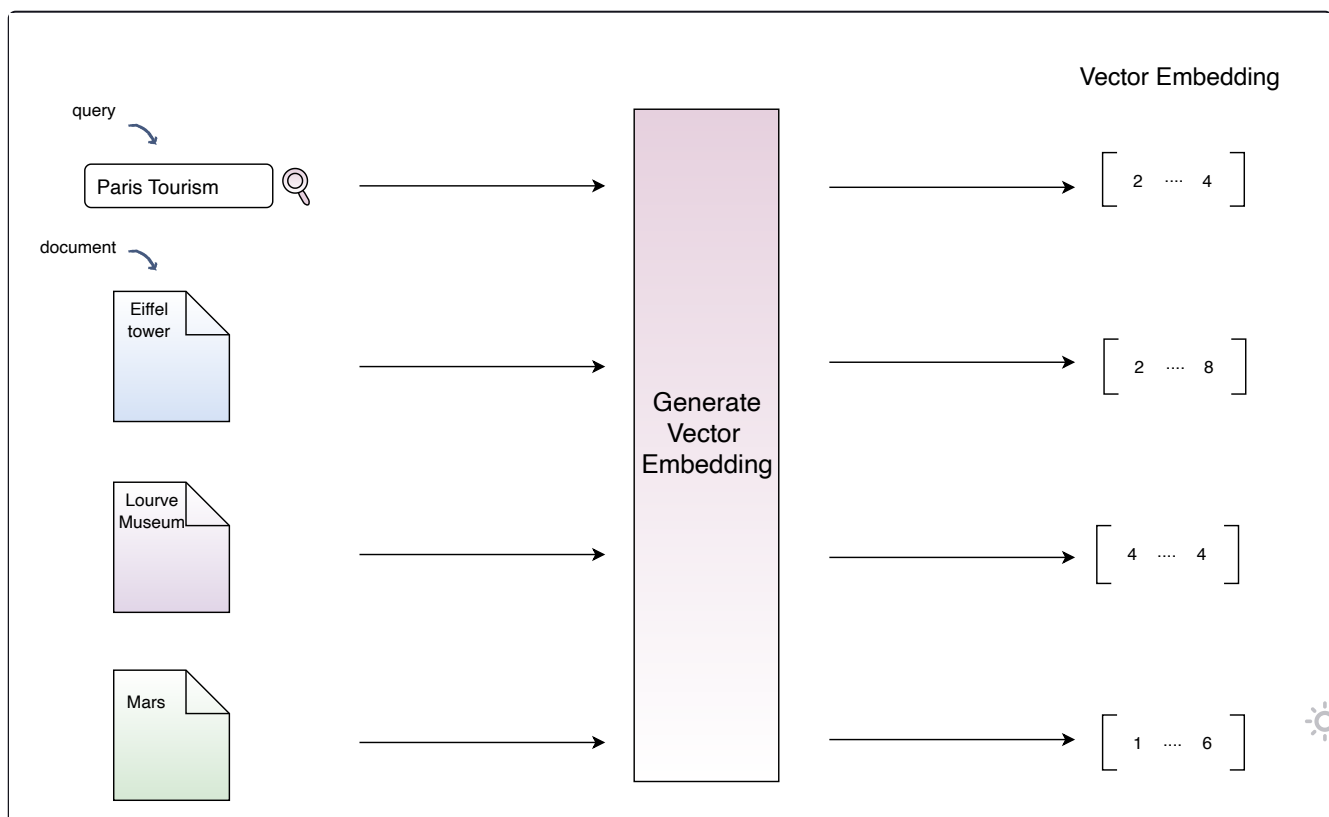


The embedding model generates vectors in such a manner that if a document is on the same topic/concept as the query, its vector is similar to the query's vector. We can use this characteristic to create a feature called “embedding similarity score”. The similarity score is calculated between the query vector and each document vector to measure its relevance for the query. The higher the similarity score, the more relevant a document is for a query.



Please refer to the [embedding lesson](#) to go over techniques that can be used to generate embeddings.

For a better understanding, let's refer to the query “Paris tourism”, as shown below.



An embedding algorithm generates vectors for the query and documents



tiv



1 of 3



Document selection selects three documents in response to the query, namely the Eiffel Tower webpage, Louvre Museum webpage, and Mars (assume this to be a website where the word “Paris” is used) webpage. We use the embeddings technique to generate vectors for the query and each of the retrieved documents. The similarity score is calculated for the query vector against each document vector. It can be seen that the Eiffel tower document has the highest similarity score. Therefore, it is the most relevant document based on semantic similarity.

Back

Next

Document Selection

Training Data Generation

☒ Mark as Completed



Report an Issue

