# Training Data Generation

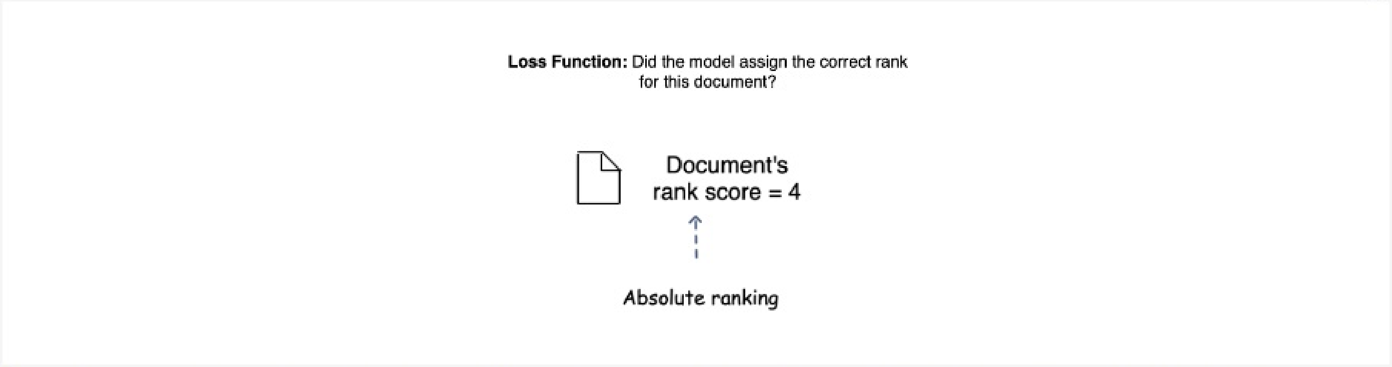
Let's look at methods for generating training data for the search ranking problem.

**We'll cover the following**

Let's generate training data for the search ranking ML model. Note that the term *training data row* and *training example* will be used interchangeably.

## Training data generation for pointwise approach

**Pointwise approach:** In this approach of model training, the training data consists of relevance scores for each document. The loss function looks at the score of one document at a time as an absolute ranking.



**Loss Function:** Did the model assign the correct rank for this document?
Document's rank score = 4
Absolute ranking

Hence the model is trained to predict the relevance of each document for a query, *individually*. The final ranking is achieved by simply sorting the result list by these document scores.

While adopting the pointwise approach, our ranking model can make use of *classification algorithms* when the score of each document takes a *small, finite number of values*. For instance, if we aim to simply classify a document as relevant or irrelevant, the relevance score will be 0 or 1. This will allow us to *approximate* the ranking problem by a **binary classification problem**.

Now let's generate training data for the binary classification approximation.

### Positive and negative training examples

We are essentially predicting user engagement towards a document in response to a query. A *relevant document* is one that successfully engages the searcher.

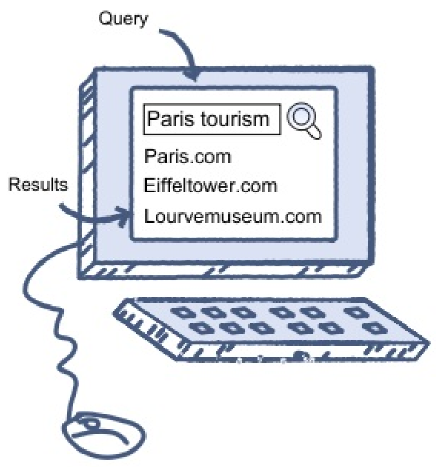
For instance, we have the searcher's query: "Paris tourism", and the following results are displayed on the SERP in response:

1. Paris.com
2. Eiffeltower.com
3. Lourvemusuem.com

We are going to label our data as positive/negative or relevant/irrelevant, keeping in mind the metric successful session rate, as shown in the following illustration.

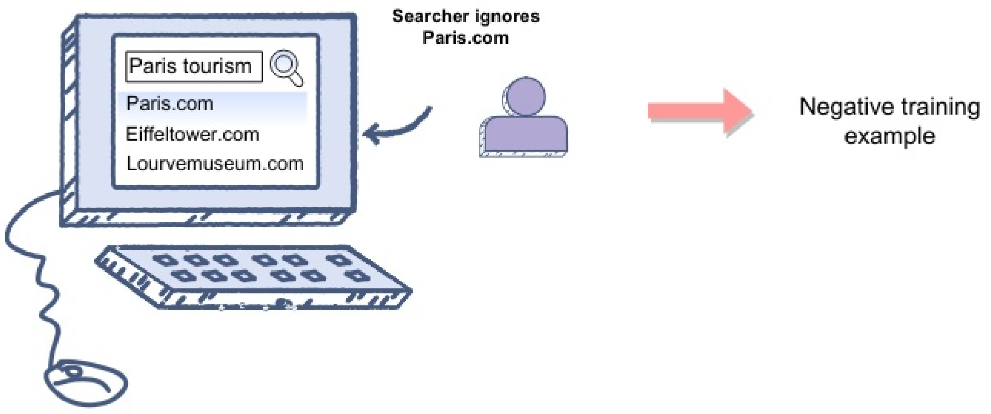
**Assumption**

Let's assume that the searcher did not engage with Paris.com but engaged with Eiffeltower.com. Upon clicking on Eiffeltower.com, they spent two minutes on the website and then signed up. After signing up, they went back to the SERP and clicked on Lourvemusuem.com and spent twenty seconds there.



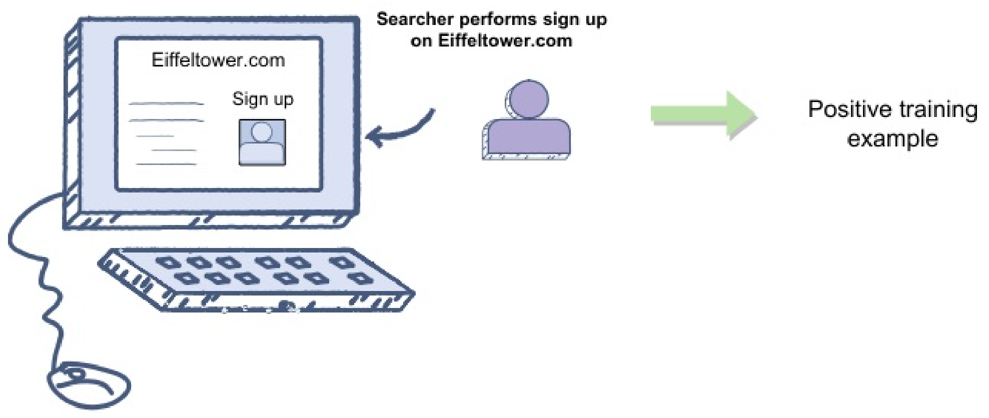
Training data generation through online user engagement

**1** of 4



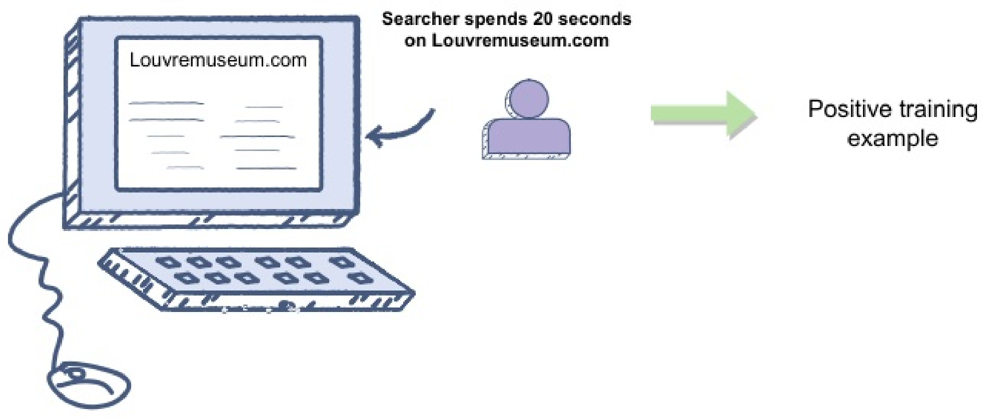
Training data generation through online user engagement

2 of 4


Searcher performs sign up on Eiffeltower.com
Eiffeltower.com
Sign up
Positive training example

Training data generation through online user engagement

3 of 4


Searcher spends 20 seconds on Louvremuseum.com
Louvremuseum.com
Positive training example

Training data generation through online user engagement

4 of 4

This sequence of events generates three rows of training data. “Paris.com” would be a negative instance because it only had a view with no engagement. The user skipped it and engaged with the other two links, which would become positive examples.

> If we had been predicting the click-through rate only, a positive example would have been one with just a click on the website.

## Caveat: Less negative examples

A question may arise that if the user engages with only the first document on the SERP, we may never get enough negative examples to train our model. Such a scenario is pretty common. To remedy it, we use *random negative* examples. For example, all the documents displayed on the $50^{th}$ page of Google search results can be considered negative examples.

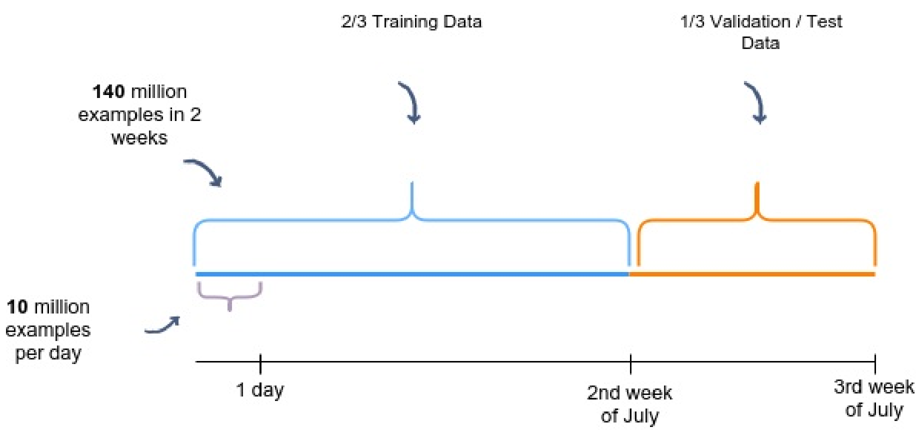
For the query discussed above, three rows of training data were generated. We may receive five million such queries per day. On average, we may generate two rows per query, one positive and one negative. This way, we would be generating ten million training examples per day.

User engagement patterns may differ throughout the week. For instance, the engagement on weekdays may differ from the weekend. Therefore, we will use a week's queries to capture all patterns during training data generation. At this rate, we would end up with around seventy million rows of training data.

## Train test split

We may randomly select $\frac{2}{3}^{rd}$, or 66.6%, of the seventy million training data rows and utilize them for training purposes. The remaining of the $\frac{1}{3}^{rd}$, or 33.3%, can be used for the validation and testing of the model.

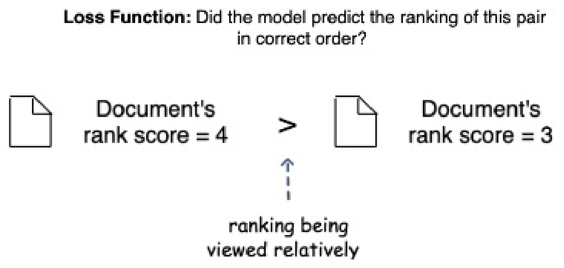
However, an approach this simple, may not be the best idea. Let's take a step back to see why. You are trying to build a model using historical data, that is trying to predict the future. You can better capture the essence of our goal by, say, training the model using data generated from the first and second week of July and data generated in the third week of July for validation and testing purposes.


2/3 Training Data
1/3 Validation / Test Data
140 million examples in 2 weeks
10 million examples per day
1 day
2nd week of July
3rd week of July

Splitting data for training, validation, and testing

## Training data generation for pairwise approach

> **Pairwise approach:** This approach differs from the pointwise approach during model training. Here, the loss function looks at the *scores of document pairs* as an inequality instead of the score of a single document. This enables the model to learn to rank documents according to their relative order which is closer to the nature of ranking.


**Loss Function:** Did the model predict the ranking of this pair in correct order?
Document's rank score = 4
>
Document's rank score = 3
ranking being viewed relatively

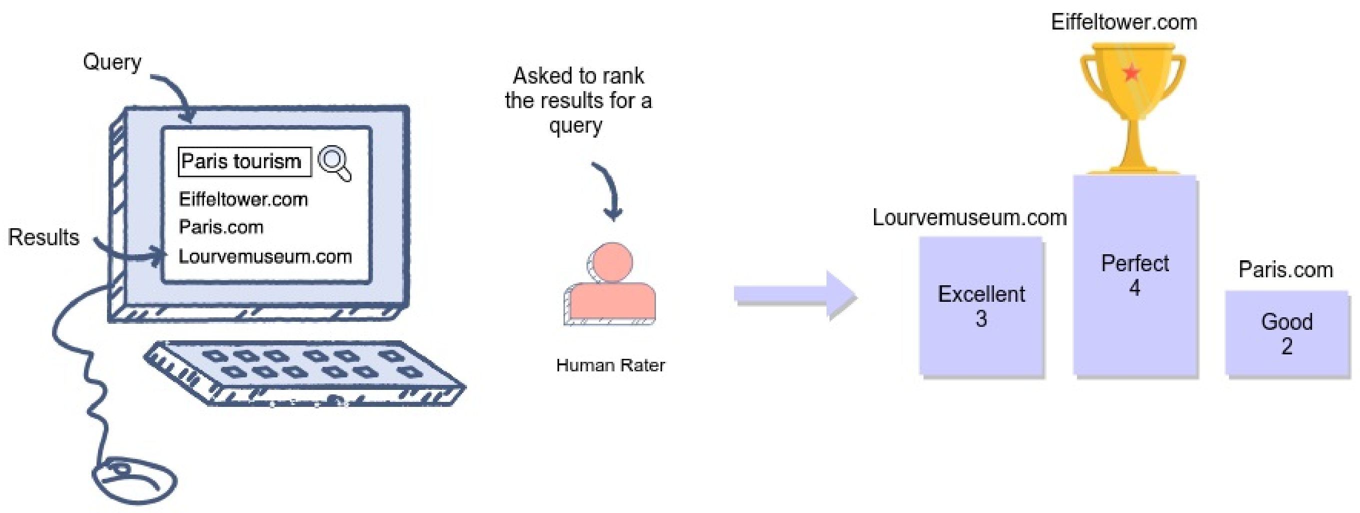
Hence, the model tries to predict the document scores such that the number of inversions in the final ranked results is minimum. Inversions are cases where the pair of results are in the wrong order relative to the ground truth.

We are going to explore two methods of training data generation for the pairwise approach to the ranking problem:

1. The use of human raters
2. The use of online user engagement

## Human raters (offline method)

In order to generate training data, human raters can help us in the following manner:



A human rater will be shown the SERP for a query, and they will rank the results

Let's assume that the human rater will be presented with $100,000$ queries, each having ten results. They will then be asked to rate the results for each query. The rater might rate a document as:

| Rating | Ranking |
|---|---|
| perfect | 4 |
| excellent | 3 |
| good | 2 |

| | |
|---|---|
| fair | 1 |
| bad | -1 |

As a result, we would have ten million training examples. The model will then learn to rank the results for a query based on these training examples.

## User-engagement (online method)

Generating a large amount of training data with the help of human raters can turn out to be very expensive. As an alternative, you can use online user engagement data to generate rankings. The user's interaction with the results on the SERP will translate to ranks based on the type of interaction.

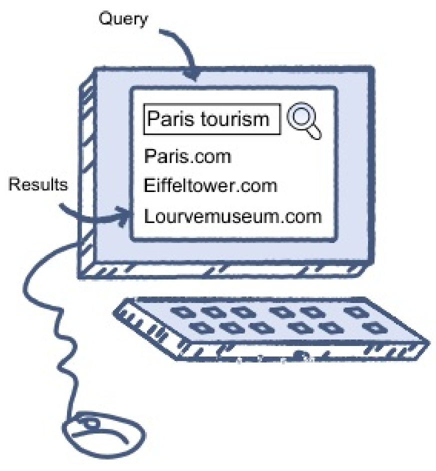
One intuitive idea can be to use online engagement to generate pairwise session data, e.g., one session can have three types of engagements varying from higher degrees of relevant actions to lower.

For instance, for the query: "Paris tourism", we get the following three results on the SERP:

1. Paris.com
2. Eiffeltower.com
3. Lourvemusuem.com

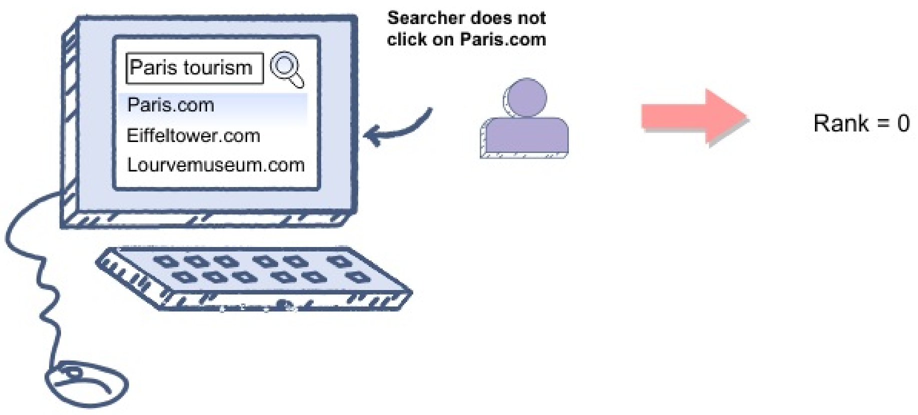
Paris.com only has an impression with no click, which would translate to label $0$. The searcher signs up on Eiffeltower.com. It will be rated as perfect and assigned the label $4$. Lourvemusuem.com will be rated as excellent and assigned the label $3$, as the searcher spends twenty seconds exploring the website.

So, this can be another way to generate pairwise data for our learning algorithm using engagement on the search results page.



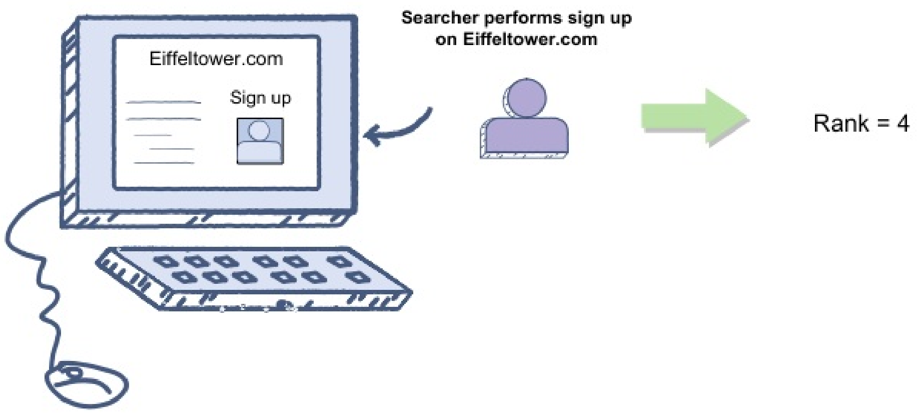
Ranking generation through online user engagement

**1** of 4



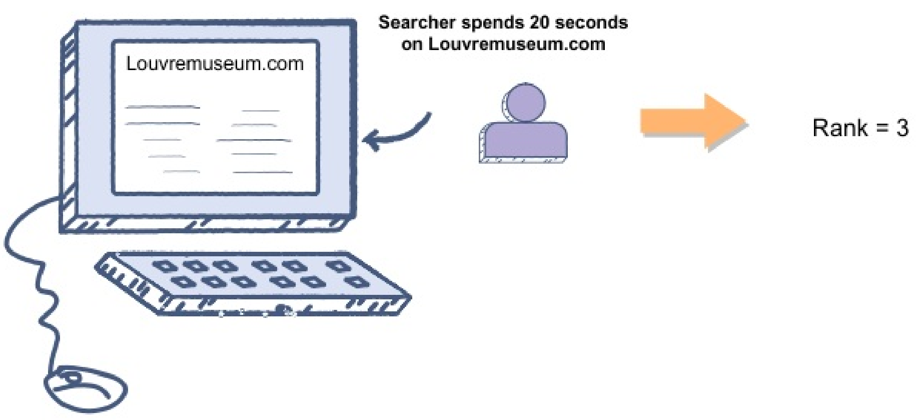
Ranking generation through online user engagement

**2** of 4



Ranking generation through online user engagement

**3** of 4



Ranking generation through online user engagement

**4** of 4