# Introduction to Search Relevance Ranking- Session I

Tutorial Link: https://dlranking.github.io/dlrr/

Data source: https://huggingface.co/datasets/xglue

XGLUE: A New Benchmark Dataset for Cross-lingual Pre-training, Understanding and Generation)

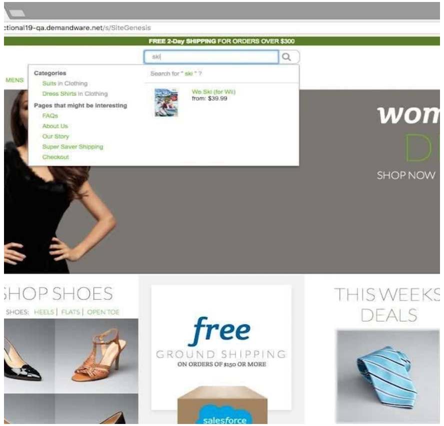
Presenters: Linsey Pang, WeiLiu, Stephen Guo

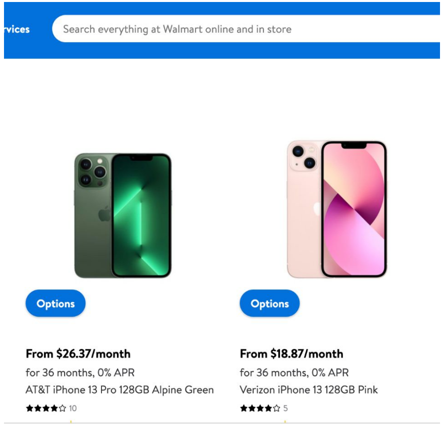
Notebooks: Linsey Pang

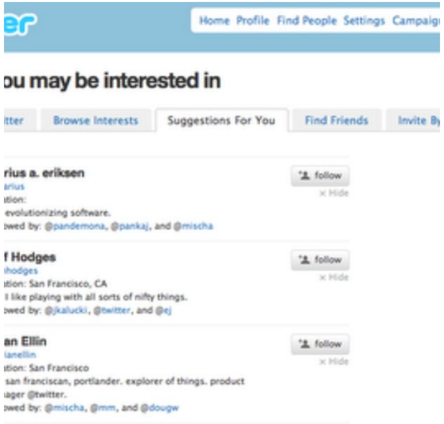
Date: August 14th, 2022

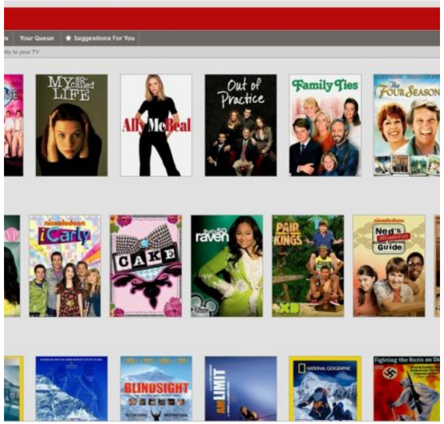
# Agenda

- Overview of search relevance ranking
- Traditional IR models
- Machine Learning approaches
- Evaluation Metrics
- Hands-on Session




**Search Engines**— Given a textual query, provide ranked list of web pages results by relevance score.

**Recommender Systems**— Given a user profile and purchase history, rank the retrieved candidates items to find personalized products for the user.

**Question Answering System** — Given a question, retrieve top answers for questions posed in natural language.

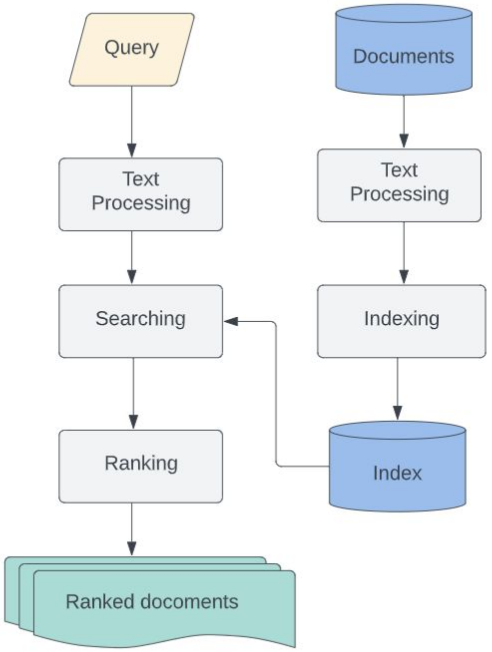
# Applications

# Information Retrieval  System-IR system

Given a query (q) and a collections (d) of documents, relevance ranking algorithms /models determine how relevant each document is for the given query.


for each input x = (q, d)  where q is a query and d is a document;

r= f(x) is relevance score function for each input.



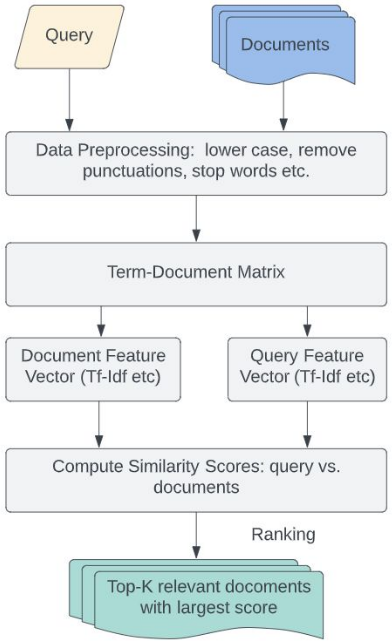
# IR Ranking Algorithms

- Vector space Model
- Probabilistic IR: BIM(the binary independence model), BM25
- Learn to Rank (machine learning approaches)

# Classical ranking Algorithms - Vector space Model

1. **What is vector space model?**

   Compute a vector embedding (e.g. using TF-IDF, Word2Vec, Doc2Vec, BERT etc ) for each query and document, and then compute the relevance score ***f*(*x*) = *f*(*q*, *d*)** as the cosine similarity between the vectors embeddings of ***q*** and ***d***.



# Classical ranking Algorithms - **Vector space Model**

## 1. Problem Statement:

Given a set of points **S** in vector space **M** and a query point **Q** ∈ ***M*,** find the closest point **S** to **Q**.

Steps:

1. Vectorize all documents – that gives **S**.
2. Vectorize the query – that gives **Q**.
3. Compute distance **D** between **Q** and **S**.
4. Sort **D** in ascending order- providing indices of most similar documents in S .
5. Return top-k of S

# Classical ranking Algorithms - Vector space Model

### 3. Demo: (TF-IDF vector feature)

- TF= term frequency is the number of times a term occurs in a document
- IDF= inverse of the document frequency, given as : IDF= log(N/df), where df is the document frequency-number of documents containing a term

For instance: total number of documents =2 ; TF matrix and IDF matrix are given:

| Terms/Docs | d1 | d2 | query |
|---|---|---|---|
| t1(red) | 1 | 1 | 0 |
| t2(dog) | 1 | 1 | 0 |
| t3(ball) | 1 | 1 | 0 |
| t4(runs) | 1 | 0 | 1 |
| t4(slow) | 0 | 1 | 0 |

| DF | IDF |
|---|---|
| 2 | 0 |
| 2 | 0 |
| 2 | 0 |
| 1 | 0.30103 |
| 1 | 0.30103 |

# Classical ranking Algorithms - Vector space Model

## 3. TF-IDF matrix

| TF-IDF Matrix | | | |
|---|---|---|---|
| Term/Docs | d1 | d2 | query |
| t1(red) | 0 | 0 | 0 |
| t2(dog) | 0 | 0 | 0 |
| t3(ball) | 0 | 0 | 0 |
| t4(runs) | 0.30103 | 0 | 0.30103 |
| t4(slow) | 0 | 0.30103 | 0 |

# Classical ranking Algorithms - Vector space Model

## 2. Similarity Metrics:

a. Cosine Similarity
b. Jaccard distance
c. Kullback-Leibler divergence
d. Euclidean distance

# Classical ranking Algorithms - Vector space Model

**3. Steps:**

a. Load documents and search queries into the R programming environment as list objects.
b. Preprocess the data by creating a corpus object with all the documents and query terms, removing stop words, punctuations using tm package.
c. Creating a term document matrix with tf-idf weight setting available in TermDocumentMatrix() method.
d. Separate the term document matrix into two parts- one containing all the documents with term weights and other containing all the queries with term weights.
e. Now calculate cosine similarity between each document and each query.
f. For each query sort the cosine similarity scores for all the documents and take top-k documents having high scores.

# Classical ranking Algorithms - BM25

**BM25 (Best Match 25)**

- Improves upon TFIDF by treating relevance as a probability problem
- Formula:

$$\text{BM25}(D, Q) = \sum_{i=1}^{n} IDF(q_i, D) \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i) + k_1 \cdot (1 - b + b \cdot |D|/d_{avg}))}$$

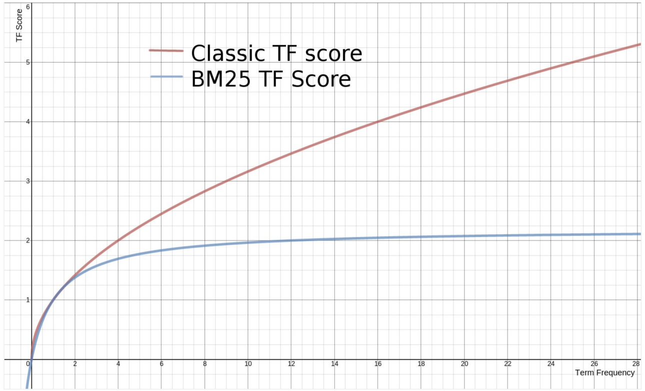
- f(qi,D) is the number of times of query term qi occurs in Document
- |D| is the number of words in document D
- D_avg is the average number of words per document
- B and k1 are hyperparameters of BM25

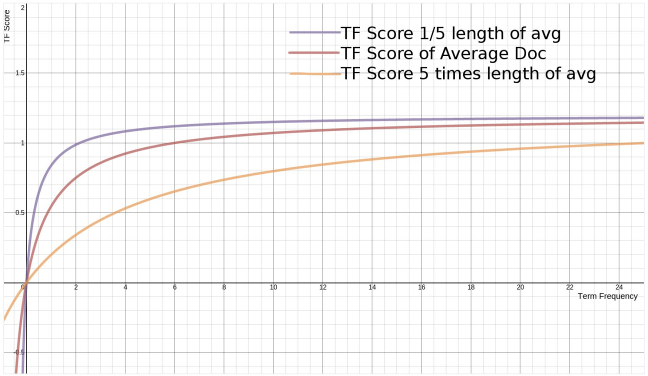
# Classical ranking Algorithms - BM25

**BM25 variables:**

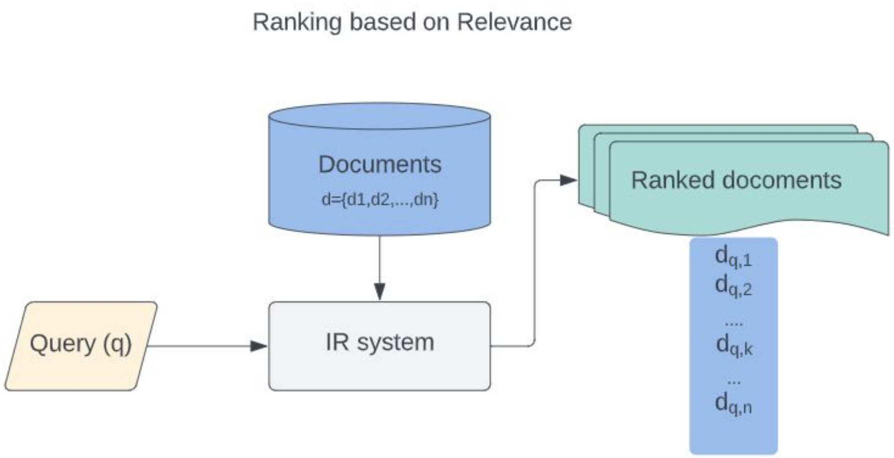
- f(qi,D) is “how many times does the ith query term occur in document D?”. The more times the query term(s) occur a document, the higher its score will be.
- k1 is a variable which helps determine TF(term frequency) saturation . The higher the value, the slower the saturation.

- |D|/d_avg : the more terms in the document that does not match input query, the lower the document's score should be.
- b (bound 0.0 ~ 1.0) : b is bigger, the effects of the document length compared to the average length are more amplified.

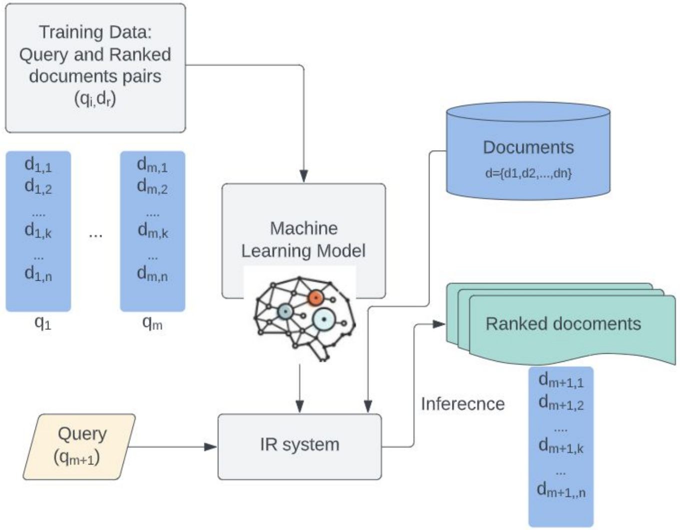


# Machine learning ranking Algorithms - overview




Traditional




Machine learning

A Short Introduction to Learning to Rank

# Machine learning ranking Algorithms - overview

**Learning to rank or Machine-learned ranking**:

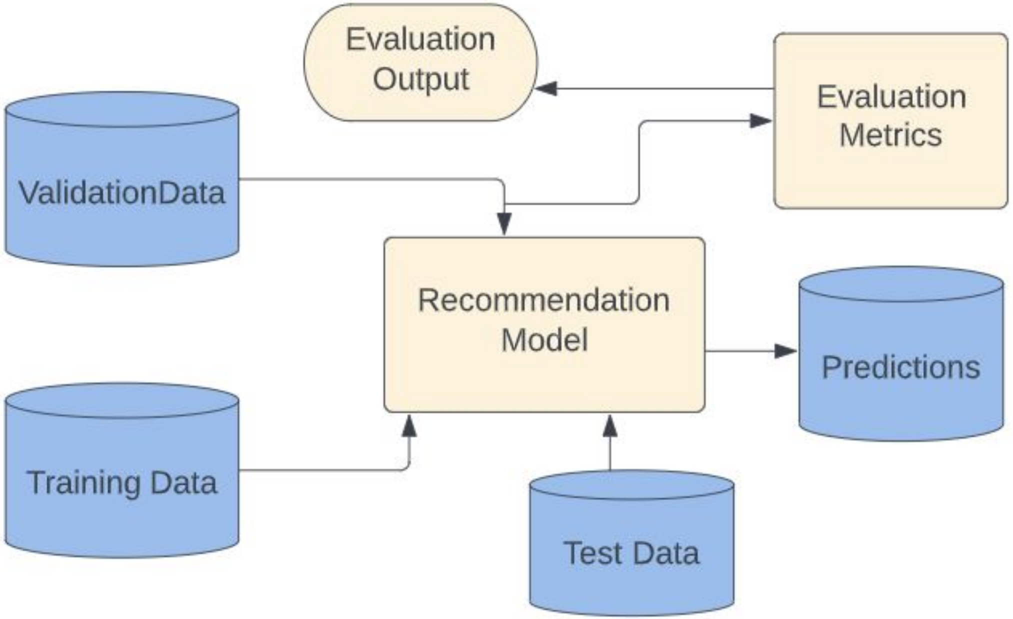
- It is  the application of machine learning, typically supervised, semi-supervised or reinforcement learning, in the construction of ranking models for information retrieval systems.

- Training data consists of lists of items with some partial order specified between items in each list. This order is typically induced by giving a numerical or ordinal score or a binary judgment (e.g. "relevant" or "not relevant") for each item.

- The goal of constructing the ranking model is to rank new, unseen lists in a similar way to rankings in the training data.

- Approaches: pointwise, pairwise, listwise
- Cost functions:

# Machine learning ranking Algorithms - **evaluation**




1. a training set is fed to a recommendation algorithm which produces a recommendation model that can be used to generate new predictions.
2. To evaluate the model a held out test set is fed to the learned model where predictions are generated for each query document pair.
3. 3. Predictions with known labels (true value) are then used as an input to the evaluation algorithm to produce evaluation results.

# Relevance Performance Metrics

- Binary assessments:
  - Precision: fraction of recommended docs that are relevant = P(relevant|recommended)
  - Recall: fraction of relevant docs that are recommended = P(recommended|relevant)

| | Relevant | NonRelevant |
|---|---|---|
| Recommended | TP | FP |
| Not-Recommended | FN | TN |

Precision = TP/(TP+FP) = # of recommendations are relevant/# of items are recommended

Recall = TP/(TP+FN) = # of recommendations are relevant/# of all possible relevant items

# Ranking Performance Metrics

- Binary relevance
  - Precision@K (P@K)
  - Recall@K(R@K)
  - F1@K(F@K)
  - Mean Average Precision (MAP)
  - Mean Reciprocal Rank (MRR)
- Multiple levels of relevance
  - Normalized Discounted Cumulative Gain (NDCG)

# Ranking Performance Metrics

Precision @k: precision evaluated only up to the  k-th prediction

$$\text{Precision@}k = \frac{true\ positives\ @k}{(true\ positives\ @k) + (false\ positives\ @k)} = \frac{\#\,of\ recommended\ items\ @k\ that\ are\ relevant}{\#\,of\ recommended\ items\ @k}$$

| k | DocID | PredictedRelevanceScore | GroudTruthRelevance (1/0) | Precision@k |
|---|---|---|---|---|
| 1 | 3 | 0.93 | 1 | 1 |
| 2 | 6 | 0.86 | 0 | 0.5 |
| 3 | 1 | 0.76 | 1 | 0.67 |
| 4 | 36 | 0.65 | 1 | 0.75 |
| 5 | 42 | 0.43 | 0 | 0.6 |
| 6 | 64 | 0.21 | 1 | 0.67 |
| 7 | 25 | 0.13 | 0 | 0.57 |

$$\text{Precision@4} = \frac{\text{true positives @4}}{(\text{true positives @4}) + (\text{false positives @4})}$$

$$= \frac{3}{3+1}$$

$$= 0.75$$

# Ranking Performance Metrics

Recall @k: Recall evaluated only up to the k-th prediction

$$\text{Recall@}k = \frac{true\ positives\ @k}{(true\ positives\ @k) + (false\ negatives\ @k)}$$

| k | DocID | PredictedRelevance Score | GroudTruthRelevance (1/0) | Recall@k |
|---|---|---|---|---|
| 1 | 3 | 0.93 | 1 | 0.25 |
| 2 | 6 | 0.86 | 0 | 0.25 |
| 3 | 1 | 0.76 | 1 | 0.5 |
| 4 | 36 | 0.65 | 1 | 0.75 |
| 5 | 42 | 0.43 | 0 | 0.75 |
| 6 | 64 | 0.21 | 1 | 1 |
| 7 | 25 | 0.13 | 0 | 1 |

$$\text{Recall@}4 = \frac{true\ positives\ @4}{(true\ positives\ @4) + (false\ negatives\ @4)}$$

$$= \frac{3}{3+1}$$

$$= 0.75$$

# Ranking Performance Metrics

F1 @k : F1 Score ranking only consider the top k prediction

$$F_1@k = 2 \cdot \frac{(Precision@k) \cdot (Recall@k)}{(Precision@k) + (Recall@k)}$$

AP(Average Precision): average of precision @k

| k | DocID | PredictedRelevanceScore | GroudTruthRelevance (1/0) | Precision@k |
|---|---|---|---|---|
| 1 | 3 | 0.93 | 1 | 1 |
| 2 | 6 | 0.86 | 0 | 0.5 |
| 3 | 1 | 0.76 | 1 | 0.67 |
| 4 | 36 | 0.65 | 1 | 0.75 |
| 5 | 42 | 0.43 | 0 | 0.6 |
| 6 | 64 | 0.21 | 1 | 0.67 |
| 7 | 25 | 0.13 | 0 | 0.57 |

$$AP = \frac{1+0.5+0.67+0.75+0.6+0.67+0.57}{7}$$
$$= 0.68$$

# Ranking Performance Metrics

MAP(Mean Average Precision): Average Precision across multiple queries/rankings; or it is a simple average of AP over all examples in a validation set.It is a simple average of AP over all examples in a validation set.

| k | GroudTruthRelevance1(1/0) | GroudTruthRelevance2(1/0) | Precision1@k | Precision2@k |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0.5 | 0.5 |
| 3 | 1 | 1 | 0.67 | 0.67 |
| 4 | 1 | 1 | 0.75 | 0.75 |
| 5 | 0 | 1 | 0.6 | 0.8 |
| 6 | 1 | 1 | 0.67 | 0.83 |
| 7 | 0 | 0 | 0.57 | 0.71 |

$$AP_1 = \frac{1+0.5+0.67+0.75+0.6+0.67+0.57}{7}$$
$$= 0.68$$
$$AP_2 = \frac{1+0.5+0.67+0.75+0.8+0.83+0.71}{7}$$
$$= 0.75$$
$$MAP = \frac{0.68+0.75}{2}$$
$$= 0.715$$

- MAP is macro-averaging: each query counts equally
- MAP assumes user is interested in finding many relevant documents for each query

# Ranking Performance Metrics

AP (average precision): measures how correct of a model's ranked prediction for a single data point

MAP(mean average precision): measures how  correct a model's ranked predictions, on average, over a whole validation set. It is computed as mean of AP over all data points in validation set.

# Ranking Performance Metrics

DCG: Discounted Cumulative Gain

$$DCG\ @k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

where reli is the relevance of the document at index i, reli equals 1 if document i is relevant and 0 otherwise.

- One advantage of DCG over other metrics is that it also works if document relevances are a real number. In other words, when each document is not simply relevant/non-relevant , but has a relevance score instead.

- Uses graded relevance as a measure of usefulness, or gain, from examining a document

Two assumptions:

- Highly relevant documents are more useful than marginally relevant documents
- The lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined

# Ranking Performance Metrics

NDCG: Normalized Discounted Cumulative Gain

$$NDCG\ @k = \frac{DCG\ @k}{IDCG\ @k}$$

$$IDCG\ @k = \sum_{i=1}^{\substack{relevant\ documents \\ at\ k}} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

*normalize* the DCG score by the maximum DCG at each threshold k

.Where IDCG @k is the best possible value for DCG @k, i.e. the value of DCG for the best possible ranking of relevant documents at threshold k

# Ranking Performance Metrics

| k | GroudTruthRank | | PreidctedRank1 | | PreidctedRank2 | |
|---|---|---|---|---|---|---|
| | DocIDs | RelevanceScore | DocIDs | RelevanceScore1 | DocIDs | RelevanceScore2 |
| 1 | 4 | 2 | 36 | 2 | 1 | 2 |
| 2 | 36 | 2 | 4 | 1 | 4 | 1 |
| 3 | 1 | 1 | 1 | 1 | 36 | 2 |
| 4 | 16 | 0 | 16 | 0 | 1 | 0 |

$$DCG = 2 + \frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} = 4.6309$$
$$DCG_1 = 2 + \frac{1}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4} = 3.6309$$
$$DCG_2 = 2 + \frac{1}{\log_2 2} + \frac{2}{\log_2 3} + \frac{0}{\log_2 4} = 4.2619$$

$$MaxDGC = 4.6309$$
$$NDCG = 4.6309/4.6309 = 1$$
$$NDCG_1 = 3.6309/4.6309 = 0.7841$$
$$NDCG_2 = 4.2619/4.6309 = 0.9203$$

# Ranking Performance Metrics

NDCG

- Normalized value
- Applicable to compare between queries
- Measure a model performance by  average NDCG values for each data point in the validation set.

# Ranking Performance Metrics: MRR

**MRR** (Mean Reciprocal Rank) :

evaluate systems that return a ranked list of answers to queries

- For a single query, the RR: *reciprocal rank* $= \frac{1}{rank}$ , where rank is position of the highest-ranked documents in (i.e. 1,2,3,...,rank, N in returned in a query)
- For multiple queries , the Mean Reciprocal Rank is the average of all queries' reciprocal ranks (RR)

- Higher MRRs mean relevant results are close to the top of search results
- Lower MRRs indicate poorer search quality, with the right answer farther down in the search results.

# XGLUE dataset

# XGLUE - QADSM dataset

QADSM: Microsoft Query-Ad Matching dataset

Dataset card | Files and versions | Community

**Dataset Preview**

Subset: qadsm

Split: train

| query (string) | ad_title (string) | ad_description (string) | relevance_label (class label) |
|---|---|---|---|
| cruise portland maine | New England Cruises | Your New England Cruise Awaits! Holland America Line Official Site. | 1 (Good) |
| transportation to cruise port miami | Holland America Line® | Explore Your World with Four Extraordinary Offers. | 0 (Bad) |
| transportation to cruise port miami | Holland America Line® | Cruise to Your Own Private Island In the Caribbean. Learn More Now. | 1 (Good) |
| galveston cruise parking | Caribbean Cruises | Sign Up for Offers and Explore the Caribbean with Holland America Line | 0 (Bad) |
| cruise portland maine | Holland America Line® | Official Site - Sign Up for Special New England Cruise Offers Today. | 1 (Good) |
| cruise portland maine | Premium Canada Cruises | Your Canada Cruise Awaits! Holland America Line Official Site. | 0 (Bad) |

# XGLUE - QADSM dataset

Dataset size:

| | | |
|---|---|---|
| Train | train | (100000, 4) |
| Validation | validation.en | (10000, 4) |
| | validation.de | (10000, 4) |
| | validation.fr | (10000, 4) |
| test | test.en | (10000, 4) |
| | test.de | (10000, 4) |
| | test.fr | (10000, 4) |

# Hands-on session

- Tutorial website: Link:
- Data Source : XGLUE: A New Benchmark Dataset for Cross-lingual Pre-training, Understanding and Generation

  Download: XGLUE (https://huggingface.co/datasets/xglue )

- Colab:
  - Vector Space Model:
  - BM25:

# Thank you!