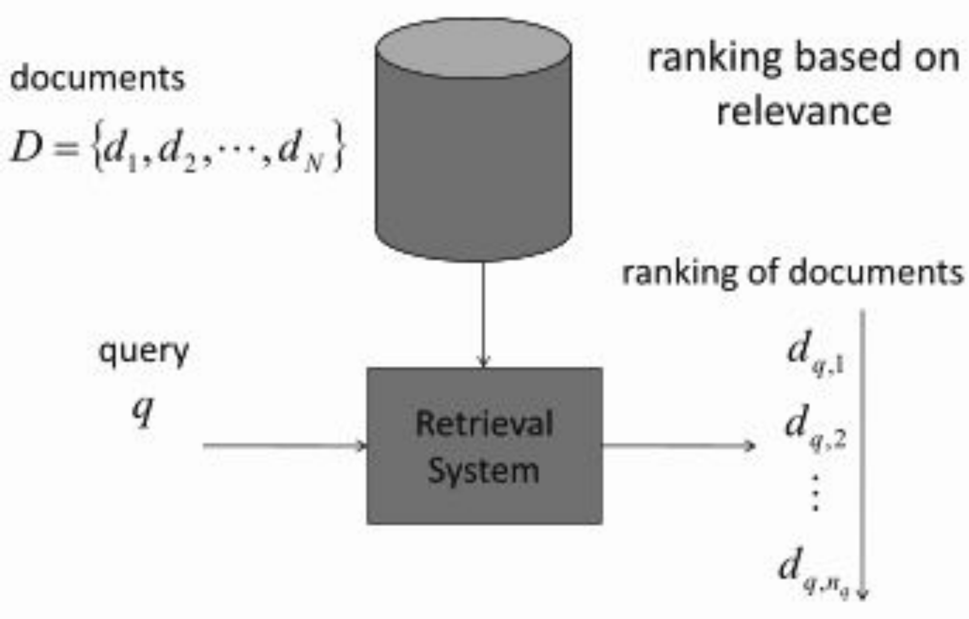
Lecture 24

- Learning to Rank

IT492: Recommendation Systems (AY 2023/24) — **Dr. Arpit Rana**

Document Retrieval



The ranking task is performed by using a ranking model *f(q, d)* to sort the documents, where **q** denotes a query and **d** denotes a document.

Learning to Rank: Motivation

- Excellent recall is insufficient for useful search; search engines also need to identify the most relevant results in a sea of matches.

- Learning to Rank algorithms aim to capture the relative utility of search results so as to return useful suggestions quickly and efficiently.

Learning to Rank: Problem Settings

Suppose $Q = \{q_1, q_2, \ldots, q_m\}$ be the set of queries for training.

For a query $\mathbf{q}_i$ -

- $D_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,n}\}$ be the set of documents associated with query $\mathbf{q}_i$
- $Y_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,n}\}$ be the set of corresponding labels, representing the relevance degree of document $\boldsymbol{d}_i$ with respect to $\mathbf{q}_i$.

So, the original training set is denoted as

$$S = \{(q_i, D_i), Y_i\}_{i=1}^{m}$$

Learning to Rank: Problem Settings

Suppose, a feature vector is created from each query-document pair **(q_i, $d_{i,j}$)**

$$x_{i,j} = \phi(q_i,\, d_{i,j}),$$
$$i = 1 \ldots m;\; j = 1 \ldots n$$

Here, ϕ denotes the feature function, i.e., features are defined as functions of a query document pair, e.g., PageRank – query dependent, query independent, and query level features.

So, we represent the training dataset as: $S = \{x_i,\, Y_i\}_{i=1}^{m}$

Where, $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}$

Learning to Rank: Problem Definition

Let the documents in D_i be identified by integers {1, 2, . . ., n}

- We define a permutation (ranking list) π_i on D_i as a *bijection* from {1, 2, . . ., n} to itself.
- We use Π_i to denote the set of all possible permutations on D_i,
- We use $\pi_i(j)$ to denote the rank (or position) of the ***j***-th document (i.e., $d_{i,j}$) in permutation π_i

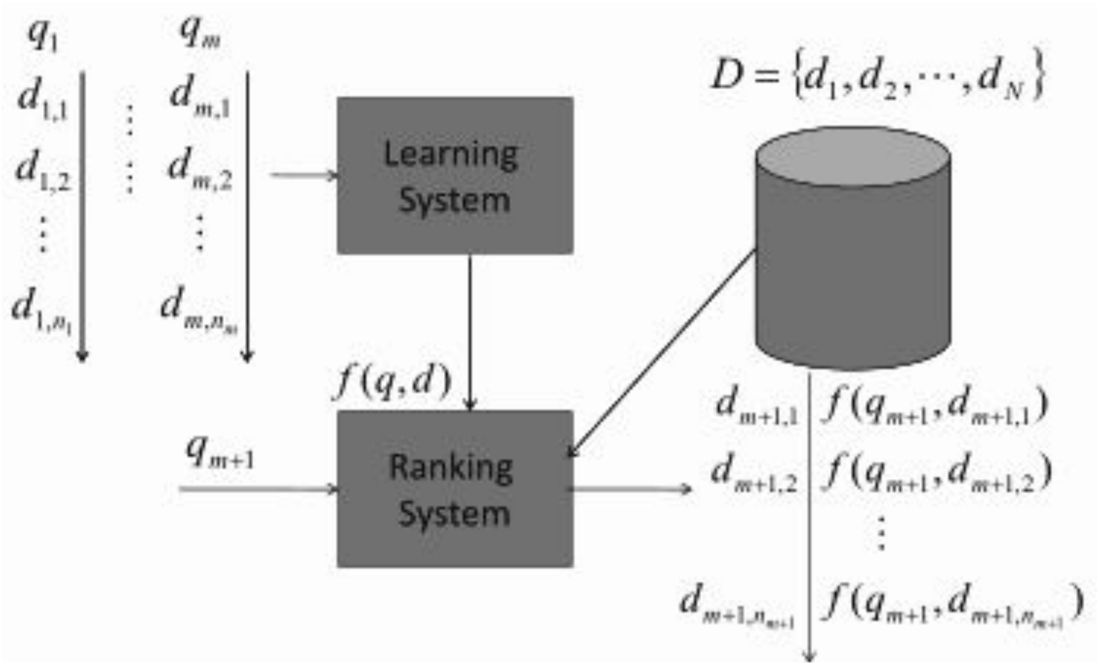
So, ranking is to select a permutation $\pi_i \in \Pi_i$ for the given query q_i and the associated documents D_i using the scores given by the ranking model $f(q_i, d_i)$.

Learning to Rank: Problem Definition

The test data T = $\{(q_{m+1}, D_{m+1, j})\}$ consists of a new query q_{m+1} and associated documents D_{m+1}.

- We create feature vector x_{m+1} ,
- use the trained ranking model to assign scores to the documents $D_{m+1, j}$,
- sort them based on the scores, and
- give the ranking list of documents as output π_{m+1}.

Learning to Rank for Document Retrieval



Learning to Rank: Evaluation

Given query q_i and associated documents D_i, suppose that π_i is the ranking list (permutation) on D_i and Y_i is the set of labels (grades) of D_i.

- DCG and NDCG measure the goodness of the ranking list with the labels at position k

The satisfaction of accessing information exponentially increases when the grade of relevance of information increases.

$$DCG(k) = \sum_{j:\pi_i(j)\leq k} \frac{2^{y_{i,j}} - 1}{\log_2(1 + \pi_i(j))},$$

The summation is taken over the top-k positions in the ranking list π_i .

$$NDCG(k) = G^{-1}_{max,i}(k) \sum_{j:\pi_i(j)\leq k} \frac{2^{y_{i,j}} - 1}{\log_2(1 + \pi_i(j))}.$$

The satisfaction of accessing information logarithmically decreases when the position of access increases.

Learning to Rank: Formulation as Supervised Learning

- Suppose that X is the input space (feature space) consisting of lists of feature vectors, and Y is the output space consisting of lists of grades.

- Let $P(X, Y)$ be an unknown joint probability distribution where random variable X takes x as its value and random variable Y takes y as its value.

- Assume that $F(\cdot)$ is a function mapping from a list of feature vectors **x** to a list of scores.

The goal of the learning task is to automatically learn a function $\tilde{F}(x)$ given training data $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$.

Learning to Rank: Formulation as Supervised Learning

Given training data, we calculate the empirical risk function as follows,

$$\hat{R}(F) = \frac{1}{m}\sum_{i=1}^{m} L(F(\mathbf{x}_i), \mathbf{y}_i).$$

The true loss function based on NDCG can be

$L(F(x), y) = 1 - NDCG$

The minimization of the empirical risk function could be difficult because it is *not continuous and it uses sorting*.

Learning to Rank: Formulation as Supervised Learning

We can consider using a surrogate loss function *L' (F(x), y)*.

$$\hat{R}'(F) = \frac{1}{m}\sum_{i=1}^{m} L'(F(\mathbf{x}_i), \mathbf{y}_i).$$

There are also different ways to define it, which lead to different approaches to learning to rank.

- For example, one can define ***pointwise loss, pairwise loss,*** and ***listwise loss functions***.

Learning to Rank: Pointwise Approach

The training data for pointwise approach consists of pairs *(x_i, y_i)*, where y_i is the relevance score of item *i*.

- The loss function used in pointwise LTR is typically a regression or classification loss such as mean squared error (MSE) or cross-entropy.

$$L'(F(\mathbf{x}), \mathbf{y}) = \sum_{i=1}^{n} (f(x_i) - y_i)^2.$$

$$L'(F(x), y) = -\left(\sum_{i=1}^{n} y_i * log(f(x_i)) + (1 - y_i) * log(1 - f(x_i))\right)$$

Learning to Rank: Pairwise Approach

The training data for pairwise approach is given as $\{((x_i^{(1)}, x_i^{(2)}), y_i)\}$, $i = 1, \cdots, m$

- where each instance consists of two feature vectors $(x_i^{(1)}, x_i^{(2)})$ and a label $y_i \in \{0, +1, -1\}$ denoting which feature vector should be ranked ahead.
- The loss function used in pairwise LTR can be the hinge loss (Ranking SVM), exponential loss (RankBoost), and logistic loss (RankNet) on pairs of objects.
- where it is assumed that $L' = 0$ when $y_i = y_j$

$$L'(F(\mathbf{x}), \mathbf{y}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \phi(\mathrm{sign}(y_i - y_j), f(x_i) - f(x_j)),$$

Learning to Rank: RankNet

- Train a feedforward neural network for each pair and predict which one is more relevant and the less relevant.
- Depending on the prediction, update the weights such that the **document which is relevant becomes even more relevant and the document which is less relevant becomes even less relevant**.

It pushes **the items at the top and at the bottom of the list equally**, whereas, it is more important to push relevant items at the top of the list.

Learning to Rank: LambdaRank

- Train a feedforward neural network for each pair and predict which one is more relevant and the less relevant.
- Depending on the prediction, update the weights such that the **document which is relevant becomes even more relevant and the document which is less relevant becomes even less relevant**.

It takes into account the order of the items in terms of NDCG based loss function.

So, **the documents at the top push more than the documents at the bottom.**

Learning to Rank: Listwise Approach

The training data for listwise LTR consists of ranked lists of features (x_1, x_2,..., x_m) extracted from query, document pairs and the corresponding ground truth relevance scores (y_1, y_2, ..., y_m).

- Listwise loss functions are defined on lists of objects, just like the true loss functions, and thus are more directly related to the true loss functions.

$$L'(F(\mathbf{x}), \mathbf{y}) = \exp(-NDCG),$$

- Methods such as AdaRank, LambdaMart have been proposed in the literature to implement listwise LTR which is a **boosted tree version of LambdaRank**.

Learning to Rank: DIY

As a part of the syllabus, you need to look at the following -

- RankNet
- LambdaRank
- LambdaMART

You can refer this document for a better understanding of the above three algorithms. Also, you can watch this conference talk by Sophie Watson.

Next Lecture

- Explanations of Recommendations