



ТЕХНОСФЕРА

Learning to Rank

Владимир Гулин, 2021 г.

План лекции

Learning To Rank

Pointwise approach

Pairwise approach

Listwise approach

Вопросы эксплуатации

Что такое “хороший” поиск?

- ▶ Находит релевантные документы
- ▶ Позволяет быстро искать
- ▶ Не находит всякий бред
- ▶ Поиск которым хочется пользоваться
- ▶ Поиск с которого переходят на “мой” сайт
- ▶ Приносит деньги

Релевантность - это способ показать насколько документ подходит запросу.

Качество ранжирования

Качество ранжирования - важнейшая характеристика поисковой системы).

На сегодняшний момент качество зависит от:

1. Оценки качества поиска.
2. Способа построения Data Set
3. Факторов поиска
4. Алгоритма обучения

Задача ранжирования

Множество запросов $Q = \{q_1, q_2, \dots, q_n\}$

Множество документов соответствующих каждому запросу $q \in Q$

$$q \rightarrow d_1, d_2, \dots$$

Для каждой пары (q, d) сопоставляется оценка релевантности $y(q, d)$, чем выше оценка, тем релевантнее документ d по запросу q .

Оценки релевантности сравнимы, только в рамках одного запроса:

$$(q, d_1) \prec (q, d_2) \iff y(q, d_1) < y(q, d_2)$$

Кренфилская методология

- ▶ Переведем ранжирование документов в последовательность чисел
- ▶ Оценим последовательности чисел
- ▶ Усредним по запросам

Как оценить ранжирование?

Дано:

- ▶ Множество запросов $Q = \{q_1, q_2, \dots, q_n\}$
- ▶ Множество документов соответствующих каждому запросу $q \in Q$.

$$q \rightarrow d_1, d_2, \dots$$

- ▶ Также для каждой пары запрос-документ имеется оценка ассесоров

Discounted Cumulative Gain

$$DCG = \sum_{i=1}^{N_q} \frac{2^{rel_i} - 1}{\log_2 i + 1}$$

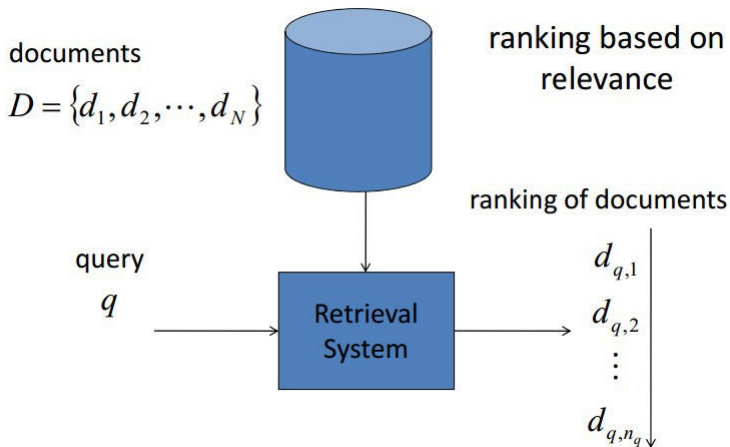
О чем мы будем говорить

Этапы ранжирования



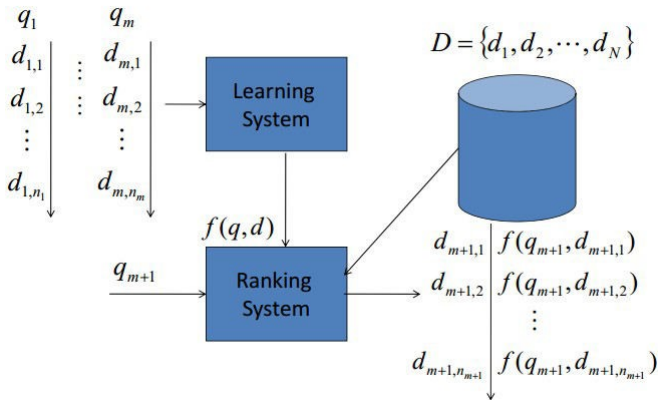
Learning to Rank

Классическое ранжирование



Learning to Rank

Ранжирование на основе машинного обучения



Факторы

Можно условно поделить на несколько видов:

- ▶ Текстовые
- ▶ Линковые
- ▶ Поведенческие
- ▶ Социальные
- ▶ Временные
- ▶ Другие
- ▶ Запросные
- ▶ Документные
- ▶ Документно-запросные
- ▶ Сайтовые
- ▶ Сайтово-запросные

Для нормального ранжирования нужно 100+ факторов.
Многие факторы сильно скоррелированы.

Алгоритмы ранжирования

Discounted Cumulative Gain

$$DCG = \sum_{i=1}^{N_q} \frac{2^{rel_i} - 1}{\log_2 i + 1}$$

Вопрос

- ▶ Как оптимизировать DCG?

Алгоритмы ранжирования

Discounted Cumulative Gain

$$DCG = \sum_{i=1}^{N_q} \frac{2^{rel_i} - 1}{\log_2 i + 1}$$

- ▶ pointwise (другая целевая функция)
 - ▶ Обучение на отдельных примерах запрос-документ
- ▶ pairwise (другая целевая функция)
 - ▶ Обучение на парах документах в рамках запроса
- ▶ listwise
 - ▶ Обучение на отранжированных списках

Algorithms

Least Square Retrieval Function (TOIS 1989) Query refinement (WWW 2008)
ListNet (ICML 2007) SVM-MAP (SIGIR 2007) Nested Ranker (SIGIR 2006)
Pranking (NIPS 2002)
LambdaRank (NIPS 2006) Frank (SIGIR 2007) MPRank (ICML 2007)
MHR (SIGIR 2007) RankBoost (JMLR 2003) Learning to retrieval info (SCC 1995)
Large margin ranker (NIPS 2002) LDM (SIGIR 2005)
RankNet (ICML 2005) Ranking SVM (ICANN 1999) IRSVM (SIGIR 2006)
Discriminative model for IR (SIGIR 2004) SVM Structure (JMLR 2005)
OAP-BPM (ICML 2003) Subset Ranking (COLT 2006)
GPRank (LR4IR 2007) QBRank (NIPS 2007) GBRank (SIGIR 2007)
Constraint Ordinal Regression (ICML 2005) McRank (NIPS 2007) SoftRank (LR4IR 2007)
AdaRank (SIGIR 2007) CCA (SIGIR 2007) ListMLE (ICML 2008)
RankCosine (IP&M 2007) Supervised Rank Aggregation (WWW 2007)
Relational ranking (WWW 2008) Learning to order things (NIPS 1998)

Pointwise approach

Идея:

Будем пытаться решать задачу ранжирования как задачу регрессии (или классификации)

$$L(h) = \sum_q \sum_{(q, d_i)} (y(q, d_i) - h(q, d_i))^2$$

- ▶ Работает!!!
- ▶ Хорошо отделяет простые запросы от сложных
- ▶ Ведет себя непредсказуемо на популярных запросах

Pointwise approach

Недостатки

- ▶ Нет непосредственной оптимизации порядка документов
- ▶ Для разных запросов разные документы будут считаться релевантными
- ▶ При переходе к задаче классификации теряется информация об упорядоченности урлов

Вопрос:

- ▶ Приведите пример, в котором будет малое значение среднеквадратичной ошибки, но плохое ранжирование.

Pairwise approach

Идея:

Будем рассматривать задачу ранжирования как задачу бинарной классификации между парами документов

Переход к гладкому функционалу качества ранжирования:

$$\begin{aligned} L(h) &= \sum_q \sum_{(q,d_i) \prec (q,d_j)} [h(\mathbf{x}_j) - h(\mathbf{x}_i) < 0] \leq \\ &\leq \sum_q \sum_{(q,d_i) \prec (q,d_j)} L(h(\mathbf{x}_j) - h(\mathbf{x}_i)) \rightarrow \min \end{aligned}$$

$h(\mathbf{x})$ - функция ранжирования;

- ▶ $L(m) = (1 - m)_+$ - RankSVM
- ▶ $L(m) = \log(1 + e^{-m})$ - RankNet
- ▶ $L(m) = e^{-m}$ - RankBoost

Ranking SVM

Линейная модель ранжирования

$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

Отступ

$$m_{i,j} = \mathbf{w}^T (\mathbf{x}_j - \mathbf{x}_i)$$

Вектор признаков

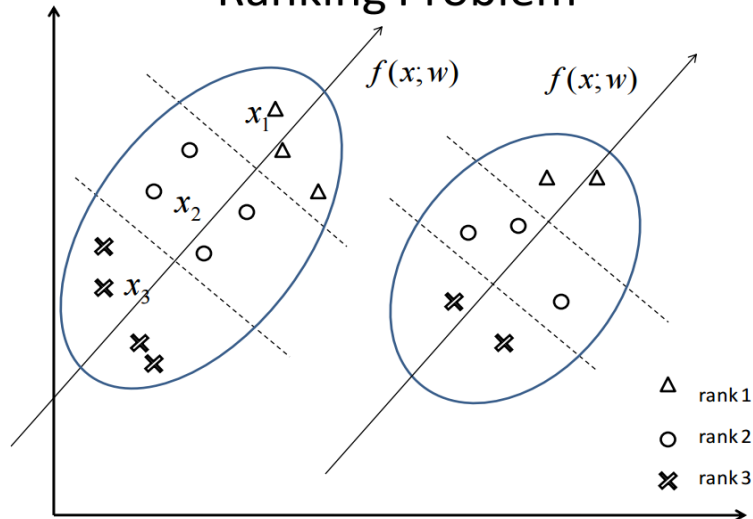
\mathbf{x}_i - описание пары запрос-документ (q, d_i)

Функционал качества

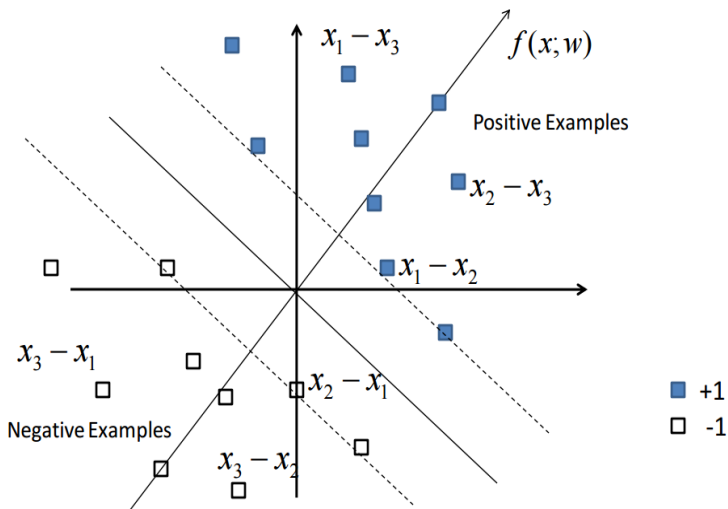
$$J(\mathbf{w}) = \sum_q \sum_{(q,d_i),(q,d_j)} (1 - m_{i,j})_+ + \frac{1}{2C} \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w}}$$

Ranking SVM

Ranking Problem



Transformed Pairwise Classification Problem



RankNet

Вероятность того, что документ d_i должен ранжироваться выше, чем документ d_j

$$P_{i,j} = P(d_j \prec d_i) = \frac{1}{1 + e^{-\sigma(h(\mathbf{x}_i) - h(\mathbf{x}_j))}}$$

Функционал качества

$$J(h) = \sum_q \sum_{(d_j \prec d_i)} -\hat{P}_{i,j} \log P_{i,j} - (1 - \hat{P}_{i,j}) \log(1 - P_{i,j})$$

$\hat{P}_{i,j}$ - “известная” вероятность того, что документ d_i должен ранжироваться выше, чем документ d_j

RankNet

Пусть $S_{i,j} \in \{0, -1, +1\}$ равно 1, если документ d_i размечен экспертами более релевантным, чем d_j , -1, если документ d_j размечен экспертами более релевантным, чем d_i , и 0, в случае равенства. Тогда

$$\hat{P}_{i,j} = \frac{1}{2}(1 + S_{i,j})$$

Можно переписать

$$J(h) = \sum_q \sum_{(d_j \prec d_i)} \frac{1}{2}(1 - S_{i,j})\sigma(h(\mathbf{x}_i) - h(\mathbf{x}_j)) + \log(1 + e^{-\sigma(h(\mathbf{x}_i) - h(\mathbf{x}_j))})$$

Дифференцируем по интересующему параметру

$$\frac{\partial J}{\partial h(\mathbf{x}_i)} = \sigma \left(\frac{1}{2}(1 - S_{i,j}) - \frac{1}{1 + e^{\sigma(h(\mathbf{x}_i) - h(\mathbf{x}_j))}} \right) = -\frac{\partial J}{\partial h(\mathbf{x}_j)}$$

RankNet

Правило обновления весов в нейросети $w_k \in \mathcal{R}$

$$w_k \rightarrow w_k - \eta \frac{\partial J}{\partial w_k} = w_k - \eta \left(\frac{\partial J}{\partial h(\mathbf{x}_i)} \frac{\partial h(\mathbf{x}_i)}{\partial w_k} + \frac{\partial J}{\partial h(\mathbf{x}_j)} \frac{\partial h(\mathbf{x}_j)}{\partial w_k} \right)$$

Перепишем

$$\begin{aligned} \frac{\partial J}{\partial w_k} &= \frac{\partial J}{\partial h(\mathbf{x}_i)} \frac{\partial h(\mathbf{x}_i)}{\partial w_k} + \frac{\partial J}{\partial h(\mathbf{x}_j)} \frac{\partial h(\mathbf{x}_j)}{\partial w_k} = \\ &= \sigma \left(\frac{1}{2}(1 - S_{i,j}) - \frac{1}{1 + e^{\sigma(h(\mathbf{x}_i) - h(\mathbf{x}_j)))}} \right) \left(\frac{\partial h(\mathbf{x}_i)}{\partial w_k} - \frac{\partial h(\mathbf{x}_j)}{\partial w_k} \right) = \\ &= \lambda_{ij} \left(\frac{\partial h(\mathbf{x}_i)}{\partial w_k} - \frac{\partial h(\mathbf{x}_j)}{\partial w_k} \right) \end{aligned}$$

где

$$\lambda_{ij} = \frac{\partial J(h(\mathbf{x}_i) - h(\mathbf{x}_j))}{\partial h(\mathbf{x}_i)} = \sigma \left(\frac{1}{2}(1 - S_{i,j}) - \frac{1}{1 + e^{\sigma(h(\mathbf{x}_i) - h(\mathbf{x}_j)))}} \right)$$

RankNet

Обзначим I набор пар индексов $\{i, j\}$, для которых $S_{ij} = 1$.
Тогда

$$\delta w_k = -\eta \sum_{\{i,j\} \in I} \left(\lambda_{ij} \frac{\partial h(\mathbf{x}_i)}{\partial w_k} - \lambda_{ij} \frac{\partial h(\mathbf{x}_j)}{\partial w_k} \right) = -\eta \sum_i \lambda_i \frac{\partial h(\mathbf{x}_i)}{\partial w_k}$$

где

$$\lambda_i = \sum_{j: \{i,j\} \in I} \lambda_{ij} - \sum_{j: \{j,i\} \in I} \lambda_{ij}$$

Смысл лямбд



LambdaRank

До сих пор оптимизировали, число неверное ранжированных пар. Как теперь оптимизировать целевую метрику (например NDCG)?

Можно показать, что

$$\lambda_{ij} = \frac{\partial J(h(\mathbf{x}_i) - h(\mathbf{x}_j))}{\partial h(\mathbf{x}_i)} = \frac{-\sigma}{1 + e^{\sigma(h(\mathbf{x}_i) - h(\mathbf{x}_j))}} |\Delta_{NDCG}|$$

где $|\Delta_{NDCG}|$ - абсолютное изменение NDCG, при обмене позиций документов U_i и U_j .

Вместо NDCG можно поставить любую другую целевую метрику.

Pairwise approach

Недостатки

- ▶ Не учитывается различное число документов для разных запросов

- Two queries in total
- Same error in terms of pairwise classification
 $780/790 = 98.73\%$.
- Different errors in terms of query level evaluation
99% vs. 50%.

		Case 1	Case 2
Document pairs of q_1	correctly ranked	770	780
	wrongly ranked	10	0
	Accuracy	98.72%	100%
Document pairs of q_2	correctly ranked	10	0
	wrongly ranked	0	10
	Accuracy	100%	0%
overall accuracy	document level	98.73%	98.73%
	query level	99.36%	50%

Известные датасеты

- ▶ LETOR 4.0
2500 запросов, 46 факторов, 3 уровня релевантности
- ▶ Internet Mathematics 2009
9124 запросов, 245 факторов, 5 уровней релевантности
- ▶ Yahoo Learning To Rank Challenge 2010
19944 запросов, 519 факторов, 5 уровней релевантности

Дано:

- ▶ Pairwise алгоритм
- ▶ Хотим оптимизировать

$$L(f(x)) = - \sum_{(i,j)} w_{ij} \log \frac{e^{f(x_i)}}{e^{f(x_i)} + e^{f(x_j)}}$$

- ▶ (i,j) - пары документов по определенному запросу
- ▶ w_{ij} - вес пары (i,j)

$$L(f(x)) = - \sum_{(i,j)} w_{ij} \log \frac{e^{f(x_i)}}{e^{f(x_i)} + e^{f(x_j)}}$$

$$dL(f(x)) = \sum_{(i,j)} w_{ij} \left((df(x_i) - df(x_j)) \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)$$

Обозначим

$$y_t = \sqrt{w_{ij}}(df(x_i) - df(x_j)), \quad a_t = \sqrt{w_{ij}} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}}$$

Будем искать y фиксированной длины

$$\begin{aligned} \arg \min_{y, |y|=const} \sum_t y_t a_t &= \arg \min_{y, |y|=const} \left(1 + 2 \sum_t \frac{y_t a_t}{|a||y|} + 1 \right) = \\ &= \arg \min_{y, |y|=const} \left(y_t + \frac{|y|}{|a|} a_t \right)^2 \end{aligned}$$

Матан

Подставим теперь выражения для y_t и a_t и определим

$$\lambda = |y|/|a|$$

$$\arg \min_{\lambda, df} \sum_{(i,j)} w_{ij} \left((df(x_i) - df(x_j)) + \lambda \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2$$

Выберем, к примеру, $\lambda = 1$

$$\arg \min_{df} \sum_{(i,j)} w_{ij} \left((df(x_i) - df(x_j)) + \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2$$

Связь с LambdaRank

В LambdaRank решение задачи упрощается

$$\arg \min_{df} \sum_{(i,j)} w_{ij} \left[\left(df(x_i) + \frac{1}{2} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2 + \left(df(x_j) - \frac{1}{2} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2 \right]$$

Если ввести обозначения

$$Val_i = \sum_j w_{ji} \frac{1}{2} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} - \sum_j w_{ij} \frac{1}{2} \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}}$$

$$W_i = \sum_j w_{ij} + \sum_j w_{ji}$$

Тогда

$$\arg \min_{df} \sum_i W_i \left(df(x_i) - \frac{Val_i}{W_i} \right)^2$$

Снова матан

Если так не делать и вспомнить что у нас специальные деревья, то можно решать задачу оптимизации сразу для пар

$$\arg \min_{df} \sum_{(i,j)} w_{ij} \left((c_{leaf(x_i)} - c_{leaf(x_j)}) + \frac{e^{f(x_j)}}{e^{f(x_i)} + e^{f(x_j)}} \right)^2$$

$$A = \begin{pmatrix} \vdots \\ 0 & 1 & -1 & 0 & \dots \\ 1 & 0 & -1 & 0 & \dots \\ \vdots \end{pmatrix}$$
$$b = \begin{pmatrix} \vdots \\ -\frac{e^{x_3}}{e^{x_2} + e^{x_3}} \\ -\frac{e^{x_3}}{e^{x_1} + e^{x_3}} \\ \vdots \end{pmatrix}$$

Отсюда методом наименьших квадратов можно найти выражение сразу для листьев

$$c = (A^T w A)^{-1} A^T w b$$

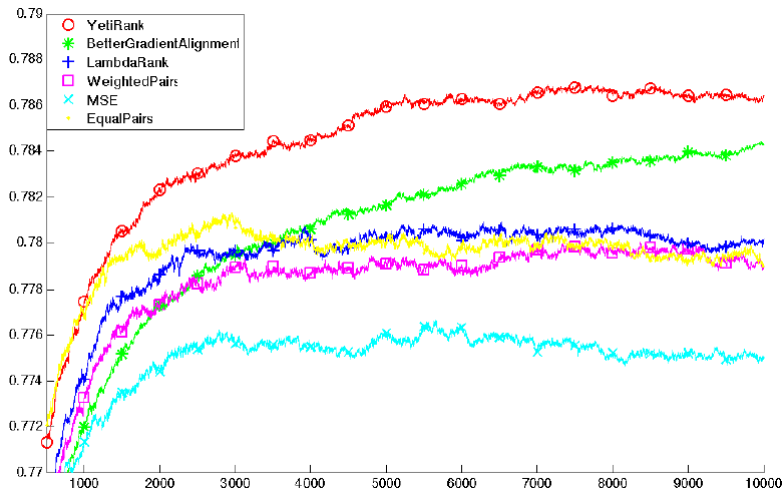
Добавим информацию об ошибках ассесоров в модель

Оценка	Нерелев.	Малополез.	Полез.	Точ. ответ	Обяз. страница
Нерелевантна	0.75	0.22	0.02	0	0
Малополезная	0.34	0.54	0.11	0.01	0
Полезная	0.07	0.13	0.73	0.06	0.01
Точный ответ	0.04	0.04	0.52	0.32	0.08
Обяз. страница	0.03	0.02	0.05	0.08	0.83

$$w_{ij} = c(l_i, l_j), \quad c(l_i, l_j) = \sum_u \sum_v I[u > v] p(u|l_i) p(v|l_j),$$

$$u, v \in 1, 2, 3, 4, 5$$

YetiRank



Listwise approach

Что все таки мешает оптимизировать NDCG напрямую?

$$NDCG = \frac{1}{NDCG_{max}} \sum_{q \in Q} \sum_{i=1}^{N_q} \frac{2^{rel_i}}{\log_2(i+1)}$$

Listwise approach

Цель:

Оптимизация целевого функционала качества ранжирования

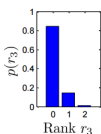
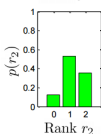
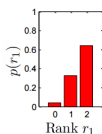
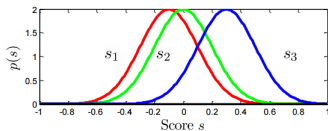
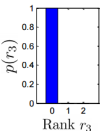
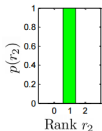
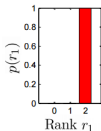
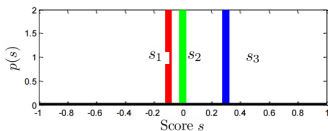
- ▶ SoftRank - сглаживание функции метрики
- ▶ AdaRank - методы бустинга
- ▶ ListNet - оптимизация гладкой функции, похожей на целевую

SoftRank

Идея:

Рассматриваем ранк каждого документа как случайную величину, распределенную по нормальному закону

$$p(s_j) = \mathcal{N}(s_j | \bar{s}_j, \sigma_s^2) = \mathcal{N}(s_j | f(\mathbf{w}, \mathbf{x}_j), \sigma_s^2)$$



Вероятностное распределение на позициях документа

Вероятность того, что i -ый документ окажется выше, чем документ j :

$$\pi_{ij} \equiv \Pr(s_i - s_j > 0) = \int_0^{\infty} \mathcal{N}(s | \bar{s}_j - \bar{s}_j, 2\sigma_s^2) ds$$

Цель:

Хочется получить вероятность того, что конкретный документ находится на позиции r

SoftRank

Рекурсивное вычисление вероятностей

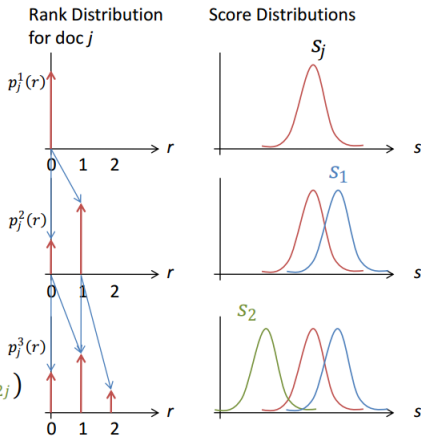
$$\pi_{ij} \equiv \Pr(s_i > s_j)$$

$$p_j^1(r) = \delta(r)$$

$$p_j^2(0) = 1 - \pi_{1j}$$

$$p_j^2(1) = \pi_{1j}$$

$$p_j^3(1) = p_j^2(0)\pi_{2j} + p_j^{i-1}(1)(1 - \pi_{2j})$$



SoftRank

► NDCG:

$$G = G_{max}^{-1} \sum_{j=1}^N g(l_j) D(r_j)$$

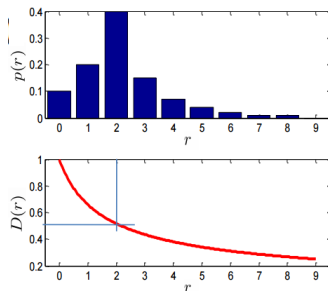
► SoftNDCG:

$$\mathcal{G} \equiv G_{max}^{-1} \sum_{j=1}^N g(l_j) E[D(r_j)]$$

$$\mathcal{G} \equiv G_{max}^{-1} \sum_{j=1}^N g(l_j) \sum_{r=0}^{N-1} D(r_j) p_j(r)$$

► Градиент:

$$\frac{\partial \mathcal{G}}{\partial \mathbf{w}} = \frac{\partial \mathcal{G}}{\partial \bar{\mathbf{s}}} \frac{\partial \bar{\mathbf{s}}}{\partial \mathbf{w}}$$



SoftRank

$$\frac{\partial \mathcal{G}}{\partial \bar{s}_m} = G_{\max}^{-1} \sum_{j=1}^N g(l_j) \sum_{r=0}^{N-1} D(r_j) \frac{\partial p_j(r)}{\partial \bar{s}_m}$$

$\frac{\partial p_j(r)}{\partial \bar{s}_m}$ вычисляем через $\frac{\partial \pi_{ij}}{\partial \bar{s}_m}$

$$\frac{\partial \pi_{ij}}{\partial \bar{s}_m} = \begin{cases} \mathcal{N}(0 | \bar{s}_m - \bar{s}_j, 2\sigma_s^2) & m = i, m \neq j \\ -\mathcal{N}(0 | \bar{s}_i - \bar{s}_m, 2\sigma_s^2) & m \neq i, m = j \\ 0 & m \neq i, m \neq j \end{cases}$$

AdaRank

Ключевые идеи:

- ▶ Аналог AdaBoost для задачи ранжирования
- ▶ Минимизирует экспоненциальную аппроксимацию функции потерь
- ▶ В качестве базовых ранкеров использует упорядочивание по значениям одного признака

Обозначения

Table 1: Notations and explanations.

Notations	Explanations
$q_i \in Q$	i^{th} query
$\mathbf{d}_i = \{d_{i1}, d_{i2}, \dots, d_{i,n(q_i)}\}$	List of documents for q_i
$y_{ij} \in \{r_1, r_2, \dots, r_\ell\}$	Rank of d_{ij} w.r.t. q_i
$\mathbf{y}_i = \{y_{i1}, y_{i2}, \dots, y_{i,n(q_i)}\}$	List of ranks for q_i
$S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$	Training set
$\vec{x}_{ij} = \Psi(q_i, d_{ij}) \in \mathcal{X}$	Feature vector for (q_i, d_{ij})
$f(\vec{x}_{ij}) \in \mathfrak{R}$	Ranking model
$\pi(q_i, \mathbf{d}_i, f)$	Permutation for q_i , \mathbf{d}_i , and f
$h_t(\vec{x}_{ij}) \in \mathfrak{R}$	t^{th} weak ranker
$E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i) \in [-1, +1]$	Performance measure function

AdaRank

$$\begin{array}{c} \max_{f \in \mathcal{F}} \sum_{i=1}^m E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i) \\ \downarrow \\ \min_{f \in \mathcal{F}} \sum_{i=1}^m (1 - E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i)) \\ \leftarrow e^{-x} \geq 1 - x \\ \downarrow \\ \min_{f \in \mathcal{F}} \sum_{i=1}^m \exp\{-E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i)\} \\ \leftarrow f(\vec{x}) = \sum_{t=1}^T \alpha_t h_t(\vec{x}) \\ \downarrow \\ \min_{h_t \in \mathcal{H}, \alpha_t \in \mathbb{R}^+} L(h_t, \alpha_t) = \sum_{i=1}^m \exp\{-E(\pi(q_i, \mathbf{d}_i, f_{t-1} + \alpha_t h_t), \mathbf{y}_i)\} \end{array}$$

AdaRank

Input: $S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$, and parameters E and T

Initialize $P_1(i) = 1/m$.

For $t = 1, \dots, T$

- Create weak ranker h_t with weighted distribution \mathbf{P}_t on training data S .
- Choose α_t

$$\alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^m P_t(i) \{1 + E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\}}{\sum_{i=1}^m P_t(i) \{1 - E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\}}.$$

- Create f_t

$$f_t(\vec{x}) = \sum_{k=1}^t \alpha_k h_k(\vec{x}).$$

- Update \mathbf{P}_{t+1}

$$P_{t+1}(i) = \frac{\exp\{-E(\pi(q_i, \mathbf{d}_i, f_t), \mathbf{y}_i)\}}{\sum_{j=1}^m \exp\{-E(\pi(q_j, \mathbf{d}_j, f_t), \mathbf{y}_j)\}}.$$

End For

Output ranking model: $f(\vec{x}) = f_T(\vec{x})$.

ListNet

- ▶ Вместо максимизации NDCG, будем минимизировать “расстояние” между истинным ранжированием и ранжированием, порождаемым ранжирующей функцией
- ▶ Метки релевантности или значения ранжирующей функции на документах порождают вероятностное распределение на множестве перестановок этих документов (модель Luce-Plackett)
- ▶ Максимизируем близость распределения, порождаемого значениями ранжирующей функции к распределению порождаемому истинными метками релевантности

ListNet

- Probability of permutation π is defined as

$$P_s(\pi) = \prod_{j=1}^n \frac{\varphi(s_{\pi(j)})}{\sum_{k=j}^n \varphi(s_{\pi(k)})}$$

- Example:

$$P_f(ABC) = \frac{\varphi(f(A))}{\varphi(f(A)) + \varphi(f(B)) + \varphi(f(C))} \cdot \frac{\varphi(f(B))}{\varphi(f(B)) + \varphi(f(C))} \cdot \frac{\varphi(f(C))}{\varphi(f(C))}$$

$P(\text{A ranked No.1})$

$P(\text{B ranked No.2} \mid \text{A ranked No.1})$
 $= P(\text{B ranked No.1}) / (1 - P(\text{A ranked No.1}))$

$P(\text{C ranked No.3} \mid \text{A ranked No.1, B ranked No.2})$

ListNet

- ▶ Модель - нейронная сеть
- ▶ Обучается градиентным спуском
- ▶ Функция потерь - KL дивергенция между распределениями, порожденными истинными метками релевантности и значениями функции ранжирования ($\phi = \exp$)

$$L(h) = - \sum_{q \in Q} \sum_{G(j_1, \dots, j_n)} \left(\prod_{t=1}^n \frac{e^{rel_{j_t}}}{\sum_{u=t}^n e^{rel_{j_u}}} \right) \log \left(\prod_{t=1}^n \frac{e^{h(x_{j_t})}}{\sum_{u=t}^n e^{h(x_{j_u})}} \right)$$

Вопрос:

- ▶ Какая есть проблема с обучением такой модели?

ListNet

- ▶ Модель - нейронная сеть
- ▶ Обучается градиентным спуском
- ▶ Функция потерь - KL дивергенция между распределениями, порожденными истинными метками релевантности и значениями функции ранжирования ($\phi = \exp$)

$$L(h) = - \sum_{q \in Q} \sum_{G(j_1, \dots, j_n)} \left(\prod_{t=1}^n \frac{e^{rel_{j_t}}}{\sum_{u=t}^n e^{rel_{j_u}}} \right) \log \left(\prod_{t=1}^n \frac{e^{h(x_{j_t})}}{\sum_{u=t}^n e^{h(x_{j_u})}} \right)$$

- ▶ Сумма по всем перестановкам $O(n!)$ в асимптотике

Top-k Probability

- ▶ При подходе “в лоб” на практике использование listnet невозможно
- ▶ Top-k Probability
 - ▶ Определим Top-k подгруппу $G(j_1, \dots, j_k)$, содержащую все перестановки, у которых top-k документов j_1, \dots, j_k

$$P_s(G(j_1, \dots, j_k)) = \prod_{t=1}^k \frac{e^{h(x_{j_t})}}{\sum_{u=t}^n e^{h(x_{j_u})}}$$

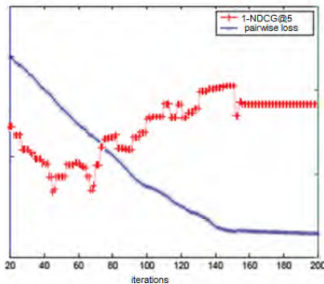
ListNet

- ▶ Модель - нейронная сеть
- ▶ Обучается градиентным спуском
- ▶ Функция потерь - KL дивергенция между **Top-k** распределениями, порожденными истинными метками релевантности и значениями функции ранжирования ($\phi = \exp$)

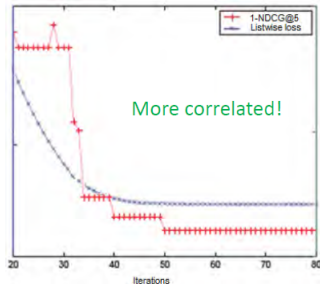
$$L(h) = - \sum_{q \in Q} \sum_{G(j_1, \dots, j_k)} \left(\prod_{t=1}^k \frac{e^{rel_{j_t}}}{\sum_{u=t}^n e^{rel_{j_u}}} \right) \log \left(\prod_{t=1}^k \frac{e^{h(x_{j_t})}}{\sum_{u=t}^n e^{h(x_{j_u})}} \right)$$

- ▶ Таким образом снижаем сложность с $O(n!)$ до $O(n!/(n-k)!)$

ListNet



Pairwise (RankNet)



Listwise (ListNet)

Training Performance on TREC Dataset

А какой подход лучше?

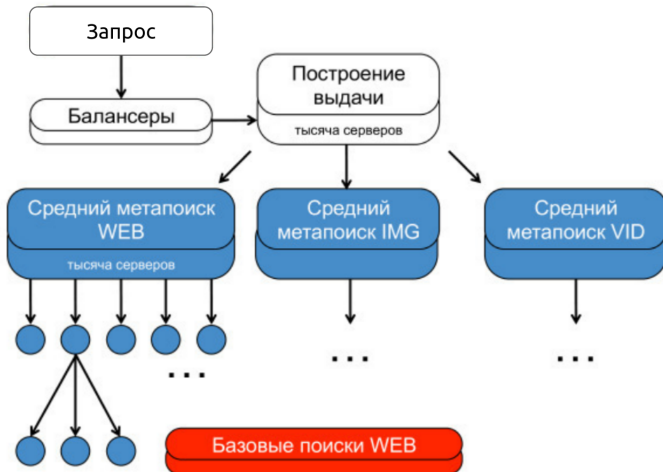
- ▶ Pointwise
- ▶ Pairwise
- ▶ Listwise

А какой подход лучше?

- ▶ Pairwise (информация пропорциональна количеству пар)
- ▶ Pointwise (информация поточечная)
- ▶ Listwise (информация позапросная)

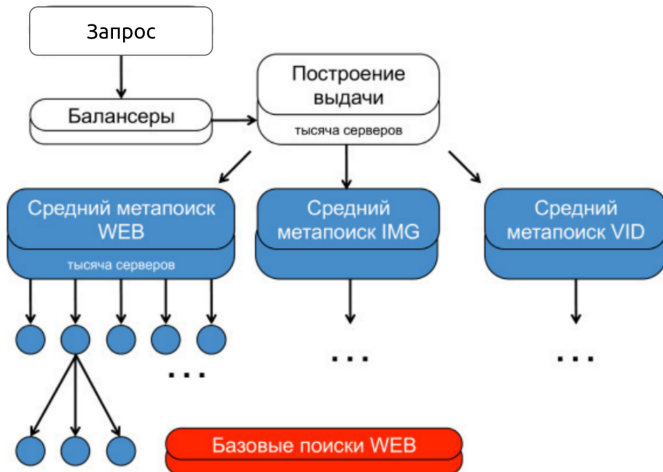
Архитектура поиска

Поиск



Архитектура поиска


Поиск



Преимущества кликов

Expert judgements	Clickthrough data
Thousands per day	Millions per day
Expensive	Cheap
Low speed of obtaining	High speed of obtaining
Noisy data	Extremely noisy data
Fresh only at the moment of assessment	Always fresh data
Can evaluate any query (not always correct)	Can't evaluate queries that nobody asks in SE
Judgements are biased	Unbiased (in terms of our flow of queries)

Кликовая добавка (pairwise)

query q	
1. url1	
2. url2	
3. url3	
4. url4	
5. url5	
6. url6	
7. url7	
8. url8	
9. url9	
10. url10	



$url2 > \begin{bmatrix} url1 \\ url3 \\ url5 \\ url6 \\ url7 \\ url9 \\ url10 \end{bmatrix}$

$url4 > \begin{bmatrix} url1 \\ url3 \\ url5 \\ url6 \\ url7 \\ url9 \\ url10 \end{bmatrix}$

$url8 > \begin{bmatrix} url1 \\ url3 \\ url5 \\ url6 \\ url7 \\ url9 \\ url10 \end{bmatrix}$

Blender (listwise)



ВОПРОСЫ ЭКСПЛУАТАЦИИ

Известные проблемы

Переобучение

- ▶ запросы;
- ▶ документы;
- ▶ эксперты;

Положительная обратная связь

- ▶ факторы;
- ▶ документы;

Шумные данные

- ▶ эксперты;

Переобучение (запросы)

Равномерная “длинная” выборка из общего лога запросов

- ▶ не обеспечивает свежести
- ▶ шумит от времени
- ▶ скачки при смене набора запросов
- ▶ оценки устаревают
- ▶ запросы становятся неактуальными

Переобучение (документы)

- ▶ Невозможно сделать равномерную выборку. Следовательно приходится учиться только на топе.
- ▶ База все время меняется. Следовательно приходится часто перестраивать модель.

Обучение на топе

- ▶ Вне топа могут встретиться совсем другие документы с аномальными для данного запроса значениями факторов
- ▶ Распределения факторов существенно смещены (Например по запросу [фк спартак] , у всех документов в заголовке есть полное вхождение данной фразы, поэтому такой фактор для данного запроса становится неинформативным)

Переобучение (документы)

- ▶ Сделать классификатор до ранжирования (например вынеси анитиспам). Однако, он ничего не будет знать про запросы.
- ▶ Сделать еще одно ранжирование, которое ставит топовые документы выше нетоповых. Однако, смещение по факторам никуда не денется.
- ▶ Активное обучение, учет неоцененных документов

Переобучение (эксперты)

- ▶ миллионы пользователей != группе экспертов
- ▶ эксперты не задают запросов
- ▶ делаем поиск не для пользователей, а для инструкции (100 стр.). Инструкция все время усложняется.

Положительная обратная связь

- ▶ поведенческие факторы
Их влияние следует ограничить частотными запросами.
- ▶ SEO. Оптимизация имеет много больше эффектов, чем кажется.
Перелинковались → ссылки стали неинформативны → все в минусе.
- ▶ документы. Надеемся на конкурентов. Добавляем новые примеры, которых нет в собственной выдаче.

Шумные данные

Что будет если перепроверить оценки?

Оценка	Нерелев.	Малополез.	Полез.	Точ. ответ	Обяз. страница
Нерелевантна	0.75	0.22	0.02	0	0
Малополезная	0.34	0.54	0.11	0.01	0
Полезная	0.07	0.13	0.73	0.06	0.01
Точный ответ	0.04	0.04	0.52	0.32	0.08
Обяз. страница	0.03	0.02	0.05	0.08	0.83

- ▶ ошибки ассесоров нужно исправлять
- ▶ попытаться учесть ошибки ассесоров при построении модели

Вопросы

