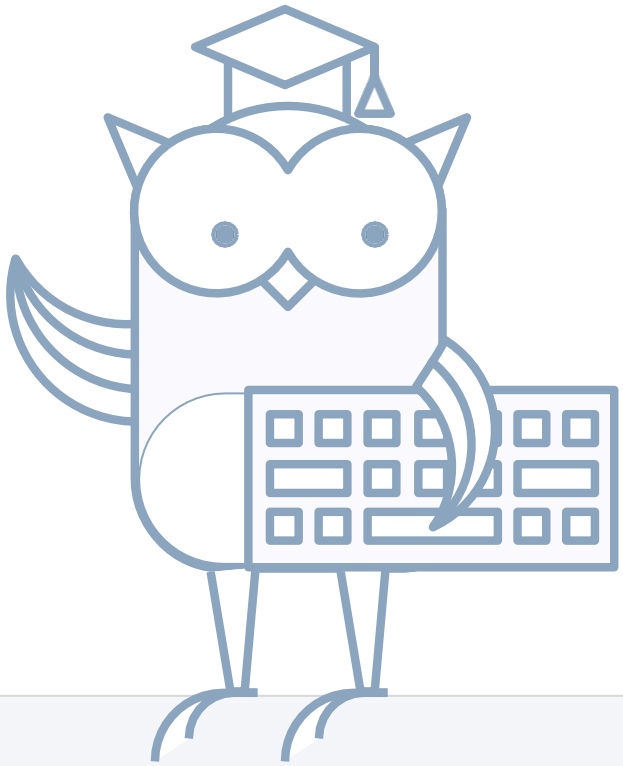
Learning to rank



Содержание

- Задача ранжирования
- Метрики
- Подходы и алгоритмы
- Существующие проблемы

Задача ранжирования

Какими качествами должен обладать хороший поиск ?

- Релевантные документы
- Быстро ищет
- Не показывает бред и мусор
- Хочется пользоваться
- Легко найти «мой» сайт
- Приносит деньги

Релевантность – мера того, на сколько документ подходит запросу

Качество ранжирования

Итоговое качество поиска зависит от:

- Оценки качества поиска

«Если вы не можете что-то измерить, то вы не можете это улучшить»

- Способа построения датасета
- Фичей
- Алгоритма

Задача ранжирования

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

Множество документов соответствующих каждому запросу $q \in Q$

$$q \rightarrow d_1, d_2, \ldots$$

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:

$$(q, d_1) < (q, d_2) \Leftrightarrow y(q, d_1) < y(q, d_2)$$

Задача ранжирования

- **Как оценить ранжирование?**

- Сопоставим нашим парам запрос-документы некоторые последовательности чисел:
  - *оценка релевантности*
  - *вероятность, что документ релевантен*
  - *релевантен/не релевантен*

- Оценим последовательности

- Усредним по запросам

Метрики. Начнем с простого

- Начнем с простого: случай бинарных ответов $y_{(i)} \in \{0,1\}$

- Любые стандартные метрики классификации:
  - *точность*
  - *полнота*
  - *F-мера*
  - *ROC-AUC*

Метрики. Начнем с простого

- Начнем с простого: случай бинарных ответов $y_{(i)} \in \{0,1\}$
- Любые стандартные метрики классификации:
  - *точность*
  - *полнота*
  - *F-мера*
  - *ROC-AUC*
- $precision@k(q)$ - точность для первых *k* предсказаний модели
- Недостаток: не учитывает порядок элементов

Метрики. Чуть посложнее

- **Average Precision** - равна сумме $p@i$ по индексам *i* от *1* до *k* только для релевантных элементов, деленому на *k.*

$$AP@k(q) = \frac{1}{k}\sum_{i=1}^{k} y_{(i)}\, precision@i(q)$$

- $y_{(i)} \in \{0,1\}$ – бинарная релевантность документа на позиции *i*
- Учитывает порядок элементов

Метрики. Чуть посложнее

- **Average Precision** - равна сумме $p@i$ по индексам *i* от *1* до *k* только для релевантных элементов, деленому на *k.*

$$AP@k(q) = \frac{1}{k}\sum_{i=1}^{k} y_{(i)}\, precision@i(q)$$

- $y_{(i)} \in \{0,1\}$ – бинарная релевантность документа на позиции *i*
- Учитывает порядок элементов

Пример:

Три элемента, релевантен последний: $ap@3(q) = \frac{1}{3}\,(0 + 0\ + 1/3) \approx 0{,}11.$

Метрики. Чуть посложнее

- **Average Precision** - равна сумме $p@i$ по индексам *i* от *1* до *k* только для релевантных элементов, деленому на *k.*

$$AP@k(q) = \frac{1}{k}\sum_{i=1}^{k} y_{(i)}\, precision@i(q)$$

- $y_{(i)} \in \{0,1\}$ – бинарная релевантность документа на позиции *i*
- Учитывает порядок элементов

Пример:

Три элемента, релевантен последний: $ap@3(q) = \frac{1}{3}\,(0 + 0\ + 1/3) \approx 0{,}11.$

Три элемента, релевантен первый: $ap@3(q) = \frac{1}{3}\,(1/1 + 0\ + 0) \approx 0{,}33.$

Метрики. Чуть посложнее

- **Average Precision** - равна сумме $p@i$ по индексам *i* от *1* до *k* только для релевантных элементов, деленому на *k.*

$$AP@k(q) = \frac{1}{k}\sum_{i=1}^{k} y_{(i)}\, precision@i(q)$$

- $y_{(i)} \in \{0,1\}$ – бинарная релевантность документа на позиции *i*
- Учитывает порядок элементов

Пример:

Три элемента, релевантен последний: $ap@3(q) = \frac{1}{3}\,(0 + 0\ + 1/3) \approx 0{,}11.$

Три элемента, релевантен первый: $ap@3(q) = \frac{1}{3}\,(1/1 + 0\ + 0) \approx 0{,}33.$

Три элемента, все релевантны: $ap@3(q) = \frac{1}{3}\,(1/1 + 2/2\ + 3/3) = 1.$

Метрики. Чуть посложнее

- **Average Precision** - равна сумме $p@i$ по индексам *i* от *1* до *k* только для релевантных элементов, деленому на *k.*

$$AP@k(q) = \frac{1}{k}\sum_{i=1}^{k} y_{(i)}\, precision@i(q)$$

- $y_{(i)} \in \{0,1\}$ – бинарная релевантность документа на позиции *i*

Метрики. Чуть посложнее

- **Average Precision** - равна сумме $p@i$ по индексам *i* от *1* до *k* только для релевантных элементов, деленому на *k.*

$$AP@k(q) = \frac{1}{k}\sum_{i=1}^{k} y_{(i)}\, precision@i(q)$$

- $y_{(i)} \in \{0,1\}$ – бинарная релевантность документа на позиции *i*

- **Mean Average Precision** – усредняем по всем запросам.

$$MAP@k = \frac{1}{N}\sum_{i=1}^{N} ap@k(q)$$

Метрики. Совсем advanced

- **DCG** *(discounted cumulative gain)*
- Ответы являются вещественными

$$\mathrm{DCG@}k(q) = \sum_{i=1}^{k} g\left(y_{(i)}\right) d(i)$$

-

Метрики. Совсем advanced

- **DCG** *(discounted cumulative gain)*
- Ответы являются вещественными

$$\mathrm{DCG@}k(q) = \sum_{i=1}^{k} g\left(y_{(i)}\right) d(i)$$

- Популярные функции:

$$g(y) = 2^{y} - 1 \qquad d(i) = \frac{1}{\log(i+1)}$$

$$\mathrm{DCG@}k(q) = \sum_{i=1}^{k} \frac{2^{y_{(i)}} - 1}{\log(i+1)}$$

Метрики. Совсем advanced

- **DCG** *(discounted cumulative gain)*
- Ответы являются вещественными

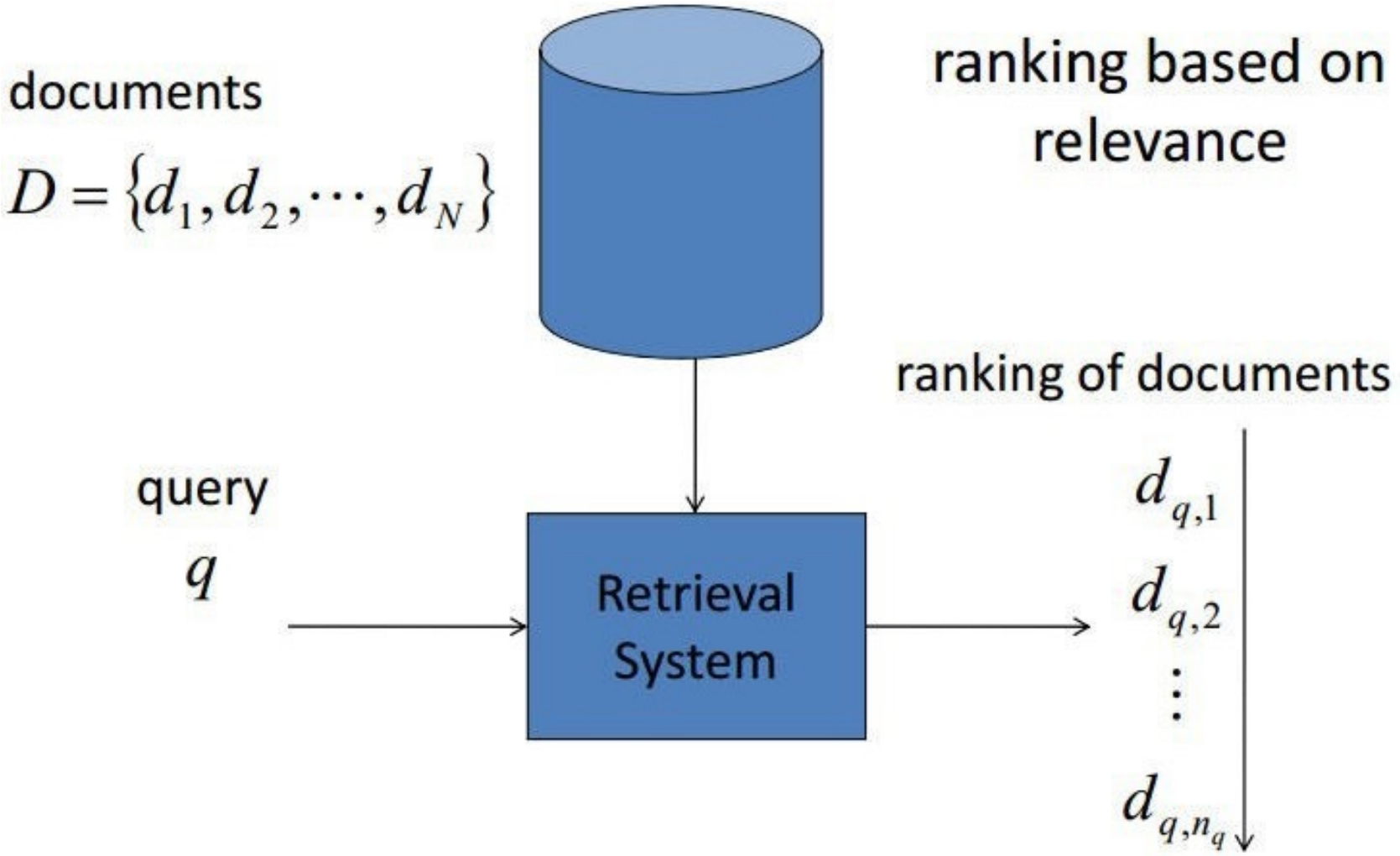
$$\mathrm{DCG@}k(q) = \sum_{i=1}^{k} \frac{2^{y_{(i)}} - 1}{\log(i+1)}$$

- **nDCG** *(normalized discounted cumulative gain)* – нормируем на идеально ранжирование

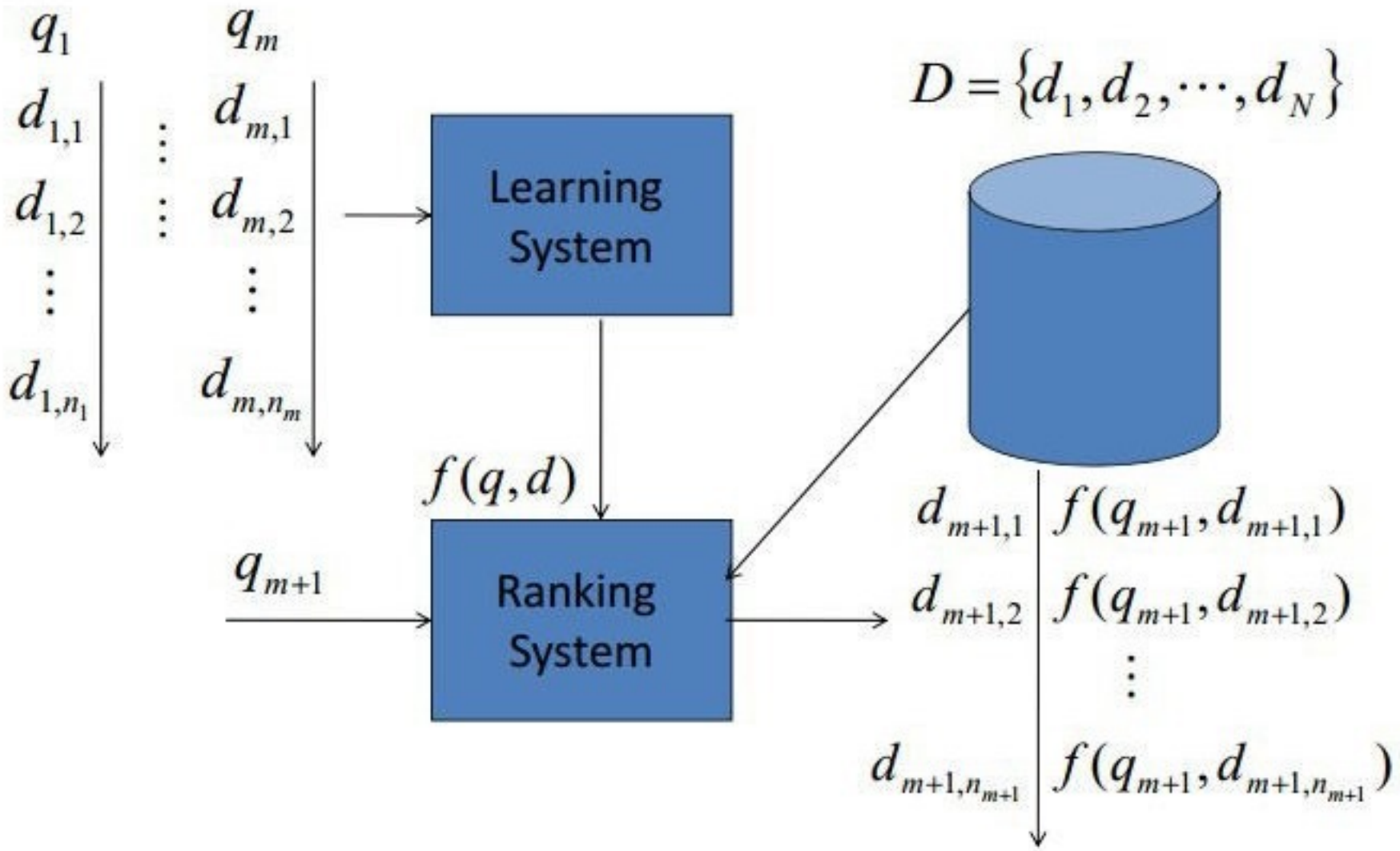
$$\mathrm{nDCG@}k(q) = \frac{\mathrm{DCG@}k(q)}{max\mathrm{DCG@}k(q)}$$

Этапы ранжирования

Классическое ранжирование



Learning to rank



Факторы ранжирования

- Текстовые
- Ссылочные
- Поведенческие
- Социальные
- Временные

Типы признаков

- Запросные:
  - популярность
  - тип
  - число слов и т. п.

Типы признаков

- Запросные:
  - популярность
  - тип
  - число слов и т. п.

- Статические (документные) – могут быть посчитаны заранее
  - популярность
  - количество ссылок
  - средний word2vec вектор и т. п.

Типы признаков

- Запросные:
  - популярность
  - тип
  - число слов и т. п.

- Статические (документные) – могут быть посчитаны заранее
  - популярность
  - количество ссылок
  - средний word2vec вектор и т. п.

- Динамические (документ – запросные)
  - расстояние между запросом и документом
  - близость тем и т. п.

Алгоритмы ранжирования

Discounted Cumulative Gain

$$\mathrm{DCG@}k(q) = \sum_{i=1}^{k} \frac{2^{y_{(i)}} - 1}{\log(i+1)}$$

Как оптимизировать?

Проблемы с дифференцируемостью….

Discounted Cumulative Gain

$$\mathrm{DCG@}k(q) = \sum_{i=1}^{k} \frac{2^{y_{(i)}} - 1}{\log(i+1)}$$

Как оптимизировать ?

Три подхода:

- ***pointwise*** (другая целевая функция), обучение на отдельных примерах
- ***pairwise*** (другая целевая функция), обучение на документах в рамках запроса
- ***listwise*** (выучиваем распределение), обучение на отранжированных списках

Pointwise

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

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

- Работает приемлемо
- Отделяет простые запросы от сложных

Недостатки

- Нет непосредственной оптимизации порядка документов
- Теряем информацию об упорядоченности
- Нет привязки к запросу
- Количество документов для запроса сильно влияет
- Нельзя влиять на конкретные результаты (YouTube по запросу)

Pairwise

Будем пытаться решить задачу ранжирования как задачу попарной классификации. Переходим к гладкому функционалу ранжирования.

$$L(h) = \sum_{q} \sum_{(q,d_i)<(q,d_j)} [h(x_j) - h(x_i) < 0]$$

$$\leq \sum_{q} \sum_{(q,d_i)<(q,d_j)} L\left(h(x_j) - h(x_i)\right) \to min$$

h(x) – функция ранжирования

Pairwise

Будем пытаться решить задачу ранжирования как задачу попарной классификации. Переходим к гладкому функционалу ранжирования.

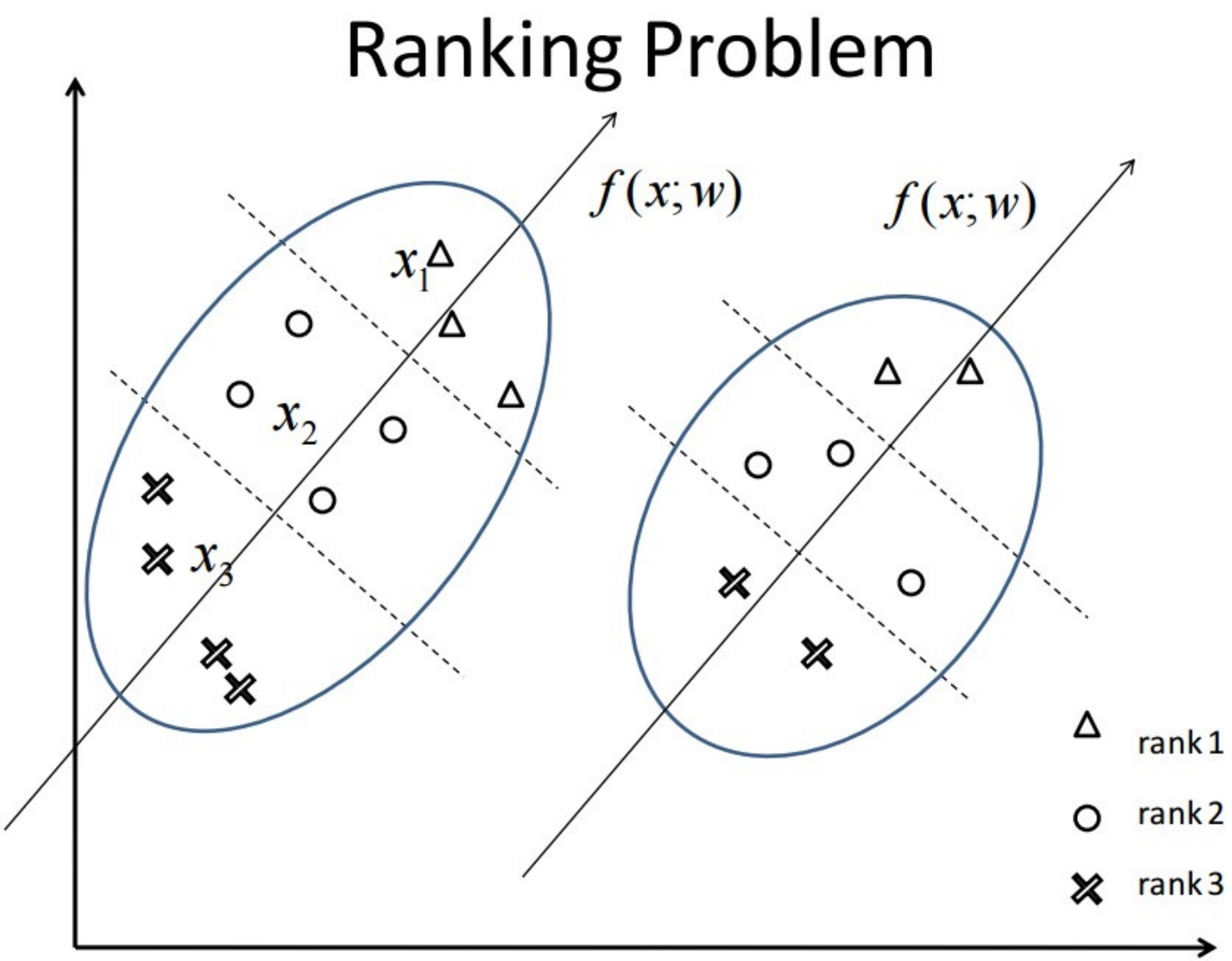
$$L(h) = \sum_q \sum_{(q,d_i)<(q,d_j)} [h(x_j) - h(x_i) < 0]$$

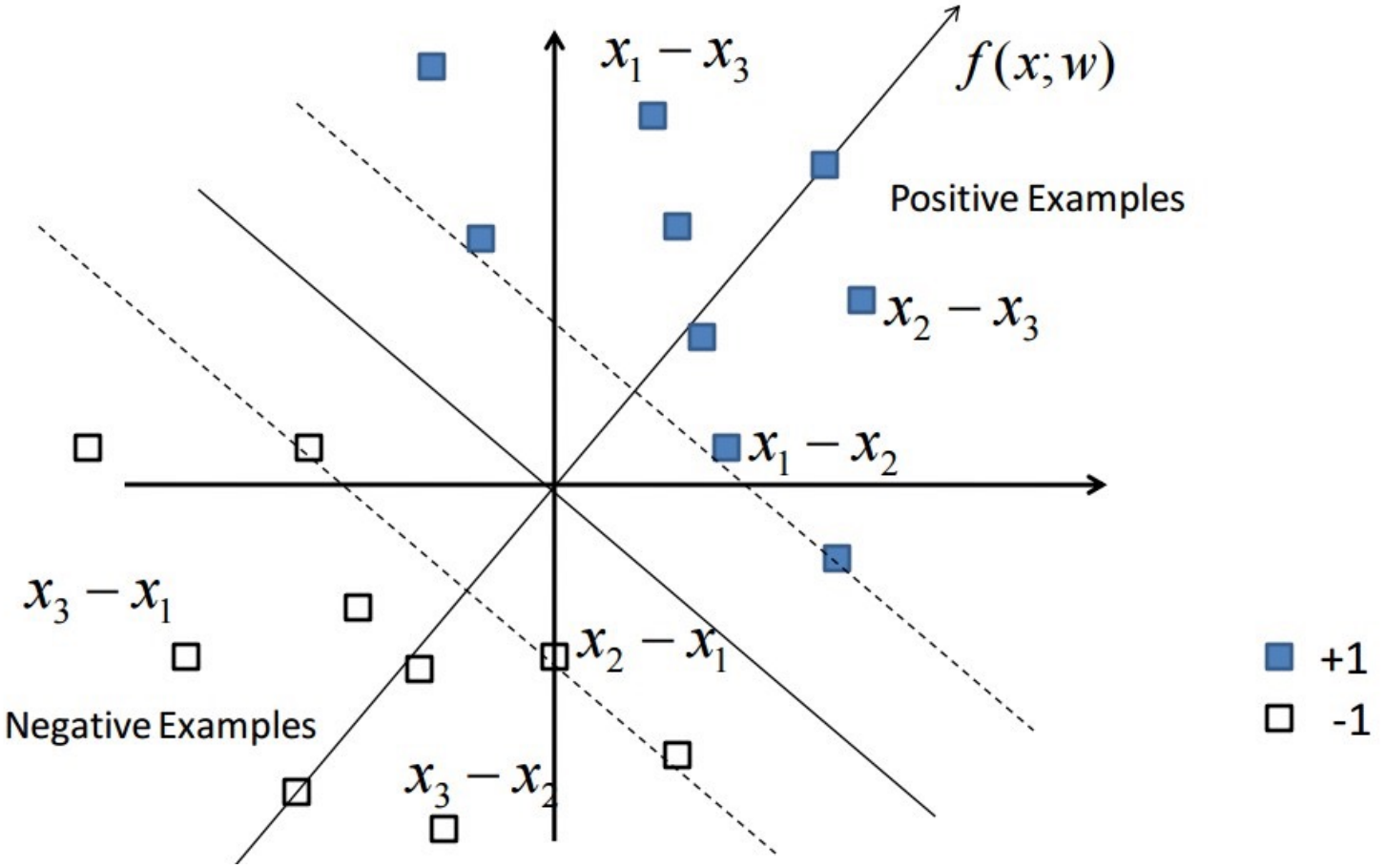
$$\leq \sum_q \sum_{(q,d_i)<(q,d_j)} L\left(h(x_j) - h(x_i)\right) \to min$$

h(x) – функция ранжирования

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

Pairwise




Transformed Pairwise Classification Problem
$x_1 - x_3$
$f(x;w)$
Positive Examples
$x_2 - x_3$
$x_1 - x_2$
$x_3 - x_1$
$x_2 - x_1$
Negative Examples
$x_3 - x_2$
+1
-1

Проблемы

OTUS

Переобучение

- запросы
- документы
- эксперты

Положительный фидбек

- факторы
- документы

Шумные данные

- эксперты

Переобучение (запросы)

Равномерная выборка из логов

- Несвежие данные
- Шумные по времени
- Скачки при смене набора запросов
- Устаревшие оценки
- Неактуальные запросы

Переобучение (документы)

Равномерная выборка из логов

- Учимся на том, что показываем
- Индекс меняется, часто перестраиваем модель
- Вне топа другое распределение данных
- Могут встречаться аномальные значения

Переобучение (эксперты)

- Пользователи != эксперты
- Асессоры не создают запросы (действуют в вакууме)
- Асессоры не эксперты в некоторых областях, могут читить
- Оценивается по большой сложной инструкции

Позитивный фидбек

- Поведенческие факторы (может быть случайным для редкого запроса)
- SEO оптимизация
- Добавляем документы, которые есть у конкурентов, но нет у нас

Шумные данные

- Асессоры тоже люди
- Пытаться учесть ошибки в модели

Listwise

- Оптимизировать целевую метрику качества вряд ли получится из-за дискретности
- But we will try to do our best!
- Введем некоторое вероятностное распределений
- Свели задачу к задаче восстановления распределения
- ListNet – списочный метод, учитывает порядок документов

ListNet

- Вместо максимизации NDCG будем минимизировать расстояние между истинным ранжированием и ранжированием, порождаемым ранжирующей функцией
- Метки релевантности или значения ранжирующей функции на документах будут порождать вероятностное распределение на перестановках
- Максимизируем близость распределения, порождаемого значениями функции и истинными метками

ListNet

- Как это работает?
- Для каждой перестановки документов π вводим вероятность этой перестановки при ранжировании нашим алгоритмом *f:*

$$P_f(\pi)$$

- Получаем два вероятностных распределения: для алгоритма (*p*) и истинное *(q)* для оценок асессоров

ListNet

- Как будем оптимизировать?
- Дивергенция Кульбаха-Лейблера!

$$\mathrm{KL}(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

- Функционал имеет смысл «расстояния между распределениями»

$$\mathrm{KL}(p\|q) \geqslant 0, \ \forall p, \ q;$$

$$\mathrm{KL}(p\|q) = 0 \iff p = q$$

- Является ли он метрикой?

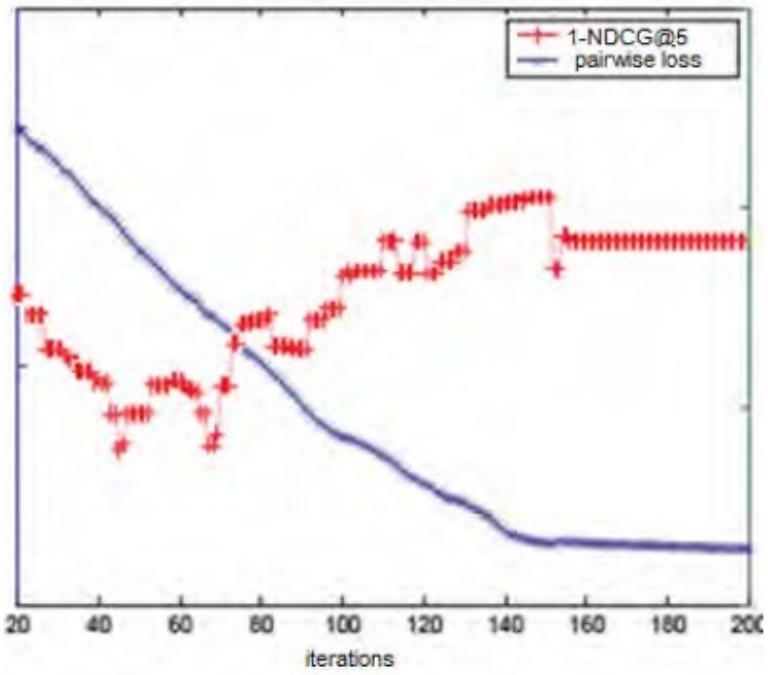
ListNet

- Используем нейронки
- Обучаем градиентным спуском
- Функция потерь – KL дифиргенция между распределениями

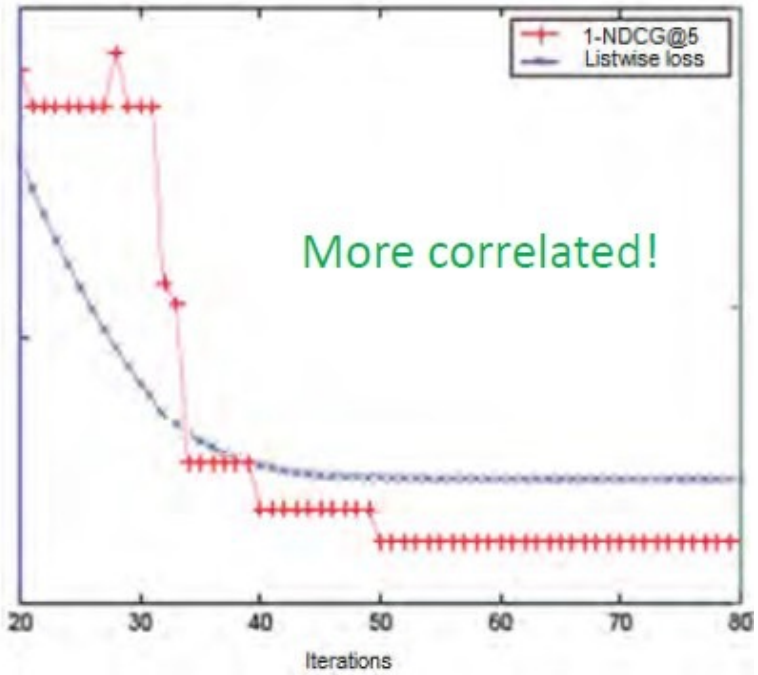
$$D_{\mathrm{KL}}(P \parallel Q) = \int_X \log \frac{\mathrm{d}P}{\mathrm{d}Q}\, \mathrm{d}P$$

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

ListNet



Pairwise (RankNet)



Listwise (ListNet)

Training Performance on TREC Dataset

Список литературы

- Статья на хабр про метрики ранжирования: https://habr.com/ru/company/econtenta/blog/303458/
- Конспект лекции Соколова ВШЭ: https://github.com/esokolov/ml-course-hse/blob/master/2019-spring/lecture-notes/lecture24-ranking.pdf
- Видео лекции Соколова: https://youtu.be/cqI3bVrFEOQ
- Лекция Воронцова http://www.machinelearning.ru/wiki/images/8/89/Voron-ML-Ranking-slides.pdf
- Статья про RankNet https://medium.com/@nikhilbd/intuitive-explanation-of-learning-to-rank-and-ranknet-lambdarank-and-lambdamart-fe1e17fac418

Спасибо
за внимание!

Bonus: Метрики. От Яндекса

- *pFound*
- Ответы на отрезке [0, 1], выражают вероятность найти ответ в документе

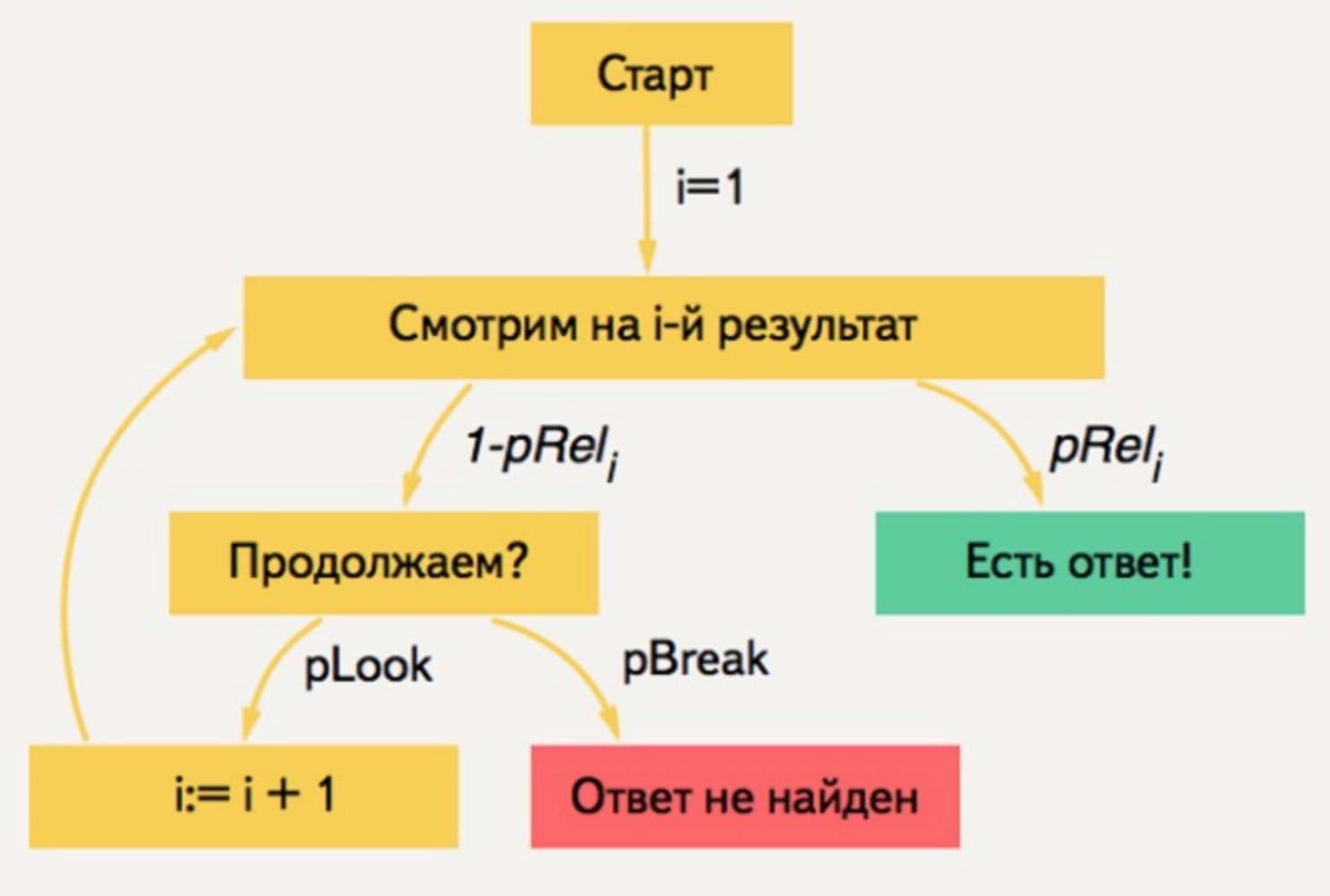
$$p_{i+1} = p_i(1 - y_{(i)})(1 - p_{out})$$

- $p_1 = 1$
- p_{out}- вероятность того, что пользователь уйдет, не найдя ответ

$$\text{pFound@}k(q) = \sum_{i=1}^{k} p_i y_{(i)}$$

- Является каскадной метрикой

Метрики. pFound



Математика за RankSVM

$$L(h) = \sum_q \sum_{(q,d_i)<(q,d_j)} [h(x_j) - h(x_i) < 0]$$

$$\leq \sum_q \sum_{(q,d_i)<(q,d_j)} L\left(h(x_j) - h(x_i)\right) \to min$$

Берем в качестве оценки сверху

$$L(m) = (1 - m)_+$$

Отступ

$$m_{i,j} = w^T(x_j - x_i)$$

В итоге получили следующую задачу минимизации:

$$L(h) = \sum_q \sum_{(q,d_i)<(q,d_j)} (1 - m_{i,j})_+ + \frac{1}{2C}\|w\|^2 \to \min_w$$

ListNet: пример функции вероятности

- 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:

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

P(A ranked No.1)

P(B ranked No.2 | A ranked No.1)
= P(B ranked No.1)/(1- P(A ranked No.1))

P(C ranked No.3 | A ranked No.1, B ranked No.2)

Проблемы

- 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%