



# ТЕХНОСФЕРА

## Learning to Rank

Владимир Гулин

29 сентября 2018 г.

# План лекции

Learning To Rank

Active Learning to Rank

# 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

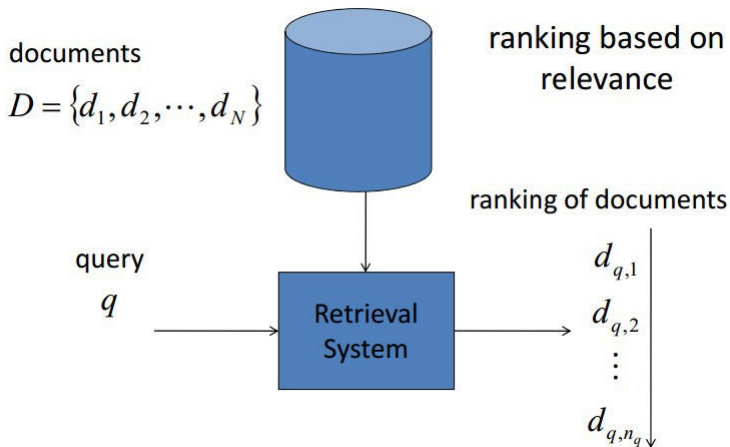
## Этапы ранжирования





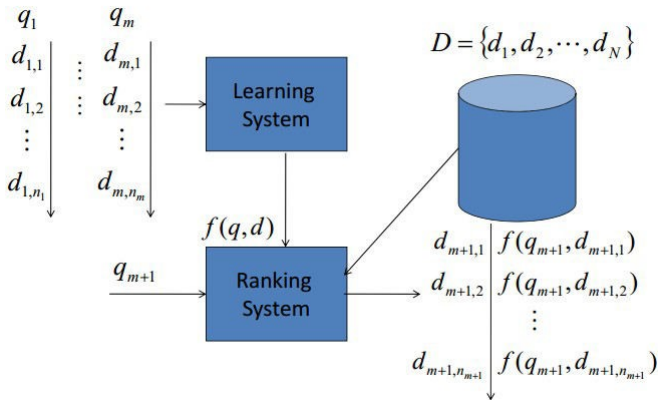
# Learning to Rank

## 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

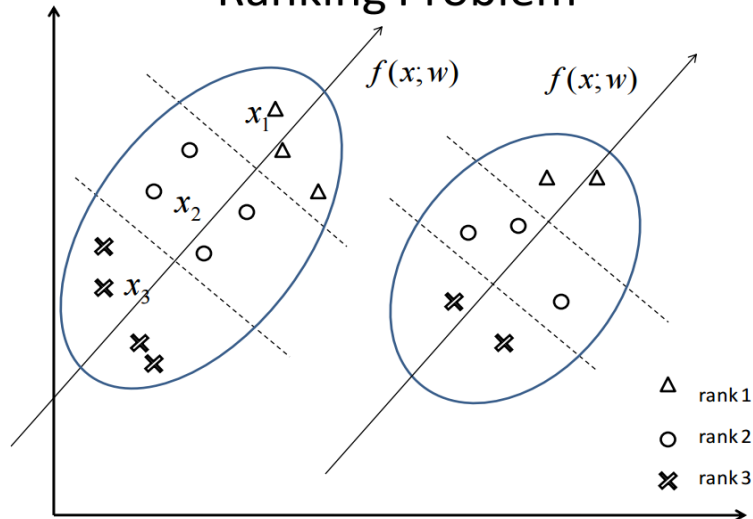
$\mathbf{x}_i$  - описание пары запрос-документ  $(q, d_i)$

Функционал качества

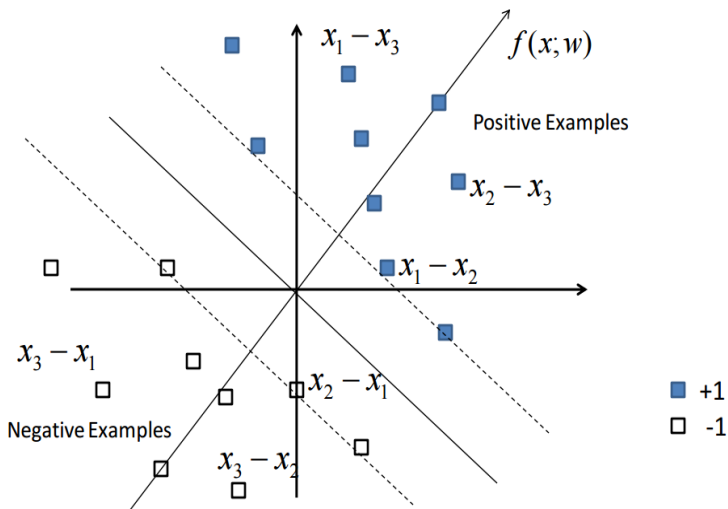
$$J(\mathbf{w}) = \sum_q \sum_{(q,d_i),(q,d_j)} (1 - m_{i,j})_+ + \frac{1}{2C} \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w}}$$

# 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

Деревья правят миром!



# 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

# ACTIVE LEARNING TO RANK

# Мотивация

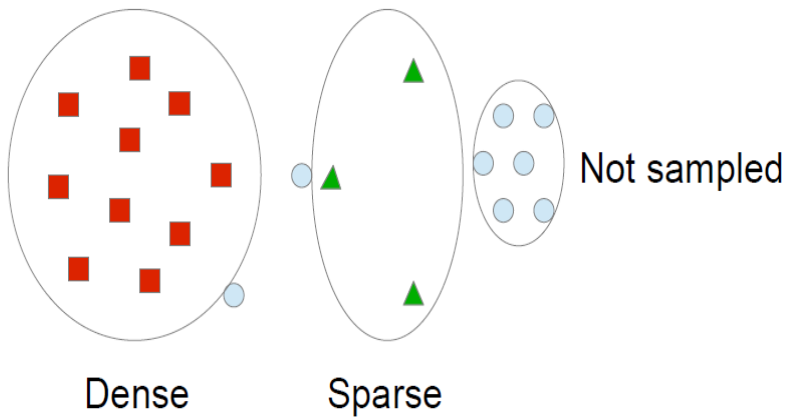
## Проблемы

- ▶ Все компоненты используют машинное обучение с учителем
- ▶ Ассесорские оценки дорогие удовольствие
- ▶ Требуются большие обучающие выборки для высокого качества
- ▶ Долго обучаться (примеров  $10^6$ )

## Хотим компактные обучающие выборки

- ▶ Проще анализировать данные
- ▶ Быстрее можно перестраивать модели и проводить эксперименты

## Density sampling

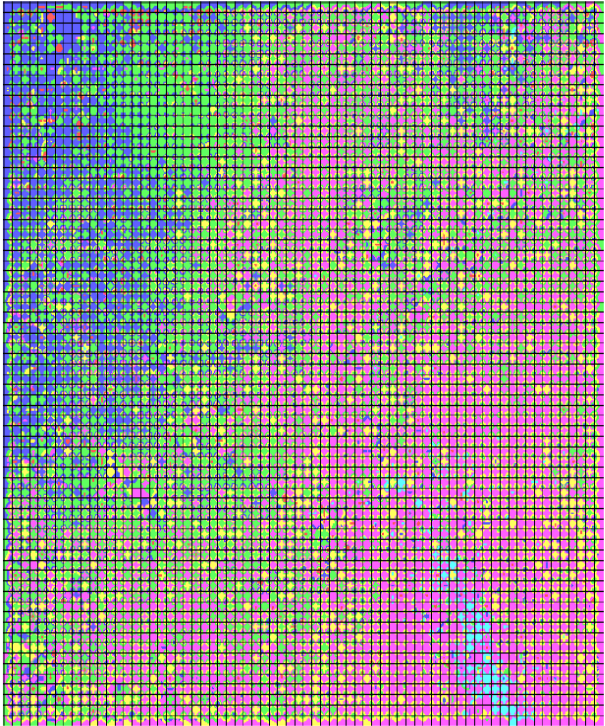


■ Points of class A

▲ Points of class B







● Unlabeled points

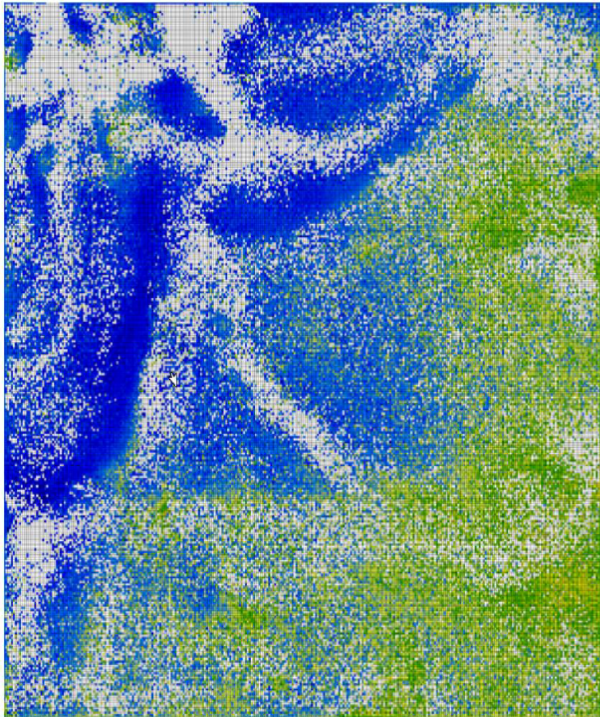




## Self-organizing map

- cell is cluster
- color is relevance

404	
Irrelevant	
Low relevant	
Medium relevant	
High relevant	
Navigation	 40 / 50



High density

Low density

# Балансировка датасета с помощью som карты

## Som balansing Algorithm

1. Build clustering  $\mathbf{C}$  for *training set*
2. Compute average density  $\mathbf{density}_{avg}$
3. For each cluster  $\mathbf{c} \in \mathbf{C}$
4. If  $\mathbf{density}(\mathbf{c}) > \mathbf{density}_{avg}$
5.     Limit number of samples in  $\mathbf{c}$  by  $\mathbf{density}_{avg}$
6. else if  $\mathbf{density}(\mathbf{c}) < \mathbf{density}_{avg}$
7.     Add  $(\mathbf{density}_{avg} - \mathbf{density}(\mathbf{c}))$  samples to cluster  $\mathbf{c}$

# Балансировка датасета с помощью som карты

## Результаты

- ▶ Training set size: 350K docs
- ▶ Map:  $300 \times 300$  clusters
- ▶ Compression: 18 %
- ▶ Quality:
  - ▶ DCG original: 17.20
  - ▶ DCG compressed: 17.26

# Query-by-Bagging

## Qbag

Input:  $T$  – initial labelled training set

$C$  – size of the committee

$A$  – learning algorithm

$U$  – set of unlabelled objects

Output:  $T'$  – extended training set

1. Uniformly resample  $T$ , obtain  $T_1 \dots T_C$ , where  $|T_i| < |T|$
2. For each  $T_i$  build model  $M_i$  using  $A$
3. Select  $x^* = \min_{x \in U} |\sum_{i=1}^C I(M_i = 1) - \sum_{i=1}^C I(M_i = 0)|$
4. Pass  $x^*$  to assessor and update  $T$
5. Repeat from 1 until convergence

## Вопрос:

- Каким образом адаптировать этот алгоритм для задачи ранжирования?

# Qbag+Som for learning to rank

## Initial training set construction algorithm

1. Build clustering using random sampling of documents
2. Mark all clusters as unused
3. Select query that covers maximum of unused clusters
4.     For each cluster covered by documents from query
5.         Label one document from this cluster
6.         Mark the cluster as used
7. Repeat from line 3 until select **M** queries

## Qbag+Som for learning to rank

### Qbag+Som Algorithm

1. Build committee of models for QBag
2. Build clustering  $\mathbf{C}$  for current training set
3. Mark all clusters as unused
4. For each query from a pool of new queries
5.     For each selected by QBag pair  $(\mathbf{d}_1, \mathbf{d}_2)$
6.          $\mathbf{c}_1 = \text{cluster}(\mathbf{d}_1)$ ,  $\mathbf{c}_2 = \text{cluster}(\mathbf{d}_2)$
7.         If  $\mathbf{c}_1$  is unused OR  $\mathbf{c}_2$  is unused
8.             Label documents  $\mathbf{d}_1$  and  $\mathbf{d}_2$
9.             Set  $\mathbf{c}_1$  and  $\mathbf{c}_2$  as used
10.     Set all clusters as unused

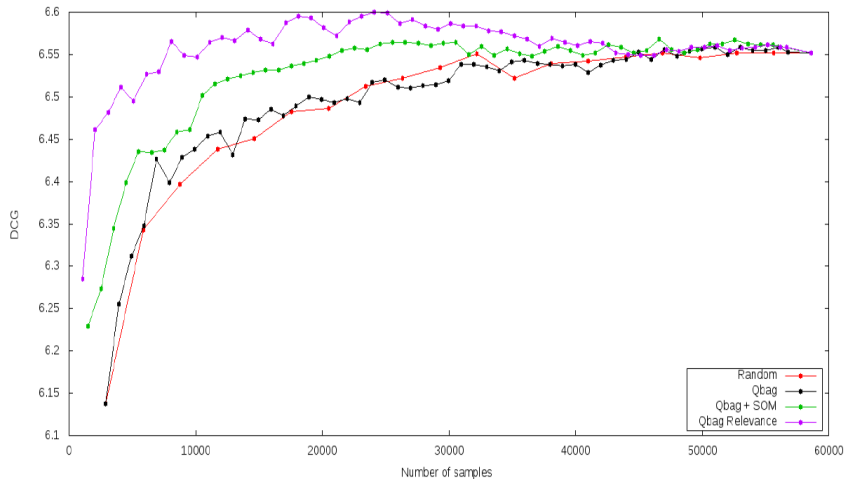
# Результаты

## Internet Mathematic 2009

- ▶ Training data: 58586 docs, 5000 queries
- ▶ Test data: 38704 docs, 4124 queries
- ▶ Size of committee: 10 models
- ▶ Size of patch at each iteration: 1000 query-docs



# Результаты



<https://yandexdataschool.com/conference/program/vlgulin>

## Домашнее задание:

### Задание:

Необходимо реализовать алгоритм ранжирования LambdaRank, в качестве целевой функции использовать NDCG. Деревья или нейросеть можно выбрать самостоятельно ) Код нужно будет прислать мне на почту для проверки.

Проверка будет реализована в виде kaggle конкурса, с отсечками по набранным баллам.

# Вопросы

