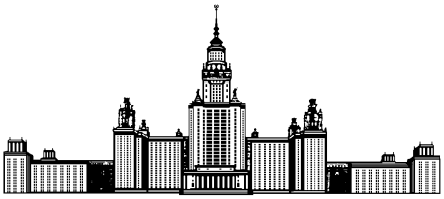

Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Алескин Александр Сергеевич

Сравнительный анализ методов решения задачи ранжирования

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
д.ф-м.н., профессор
В.В. Рязанов

Москва, 2017

Содержание

Аннотация

В данной работе проведен сравнительный анализ существующих методов ранжирования для текстовых документов. Было предложено два метода для повышения качества ранжирования. Первый алгоритм заключается в сглаживании рейтингов документов через рейтинги соседних документов. Цель второго заключается в добавлении дополнительных признаков в виде тематик документа. Были проведены эксперименты, которые подтверждают стабильное улучшение работы методов ранжирования при использовании предложенных алгоритмов.

1 Введение

Сегодня, в связи с быстрым развитием электроники и интернета, доступность информации для человека значительно увеличилась. Однако, при этом существенно возросли и её объёмы: что сделало весьма сложной и актуальной задачу поиска требуемой информации. Один из способов решить данную проблему — использовать специальные методы ранжирования, которые позволяют отсортировать необходимую информацию.

Задача для данных алгоритмов ставиться следующим образом. Даны поисковый запрос пользователя и список документов, которые надо отранжировать в таком порядке, чтобы вначале шли документы с информацией, которая с большой степенью релевантна к поисковому запросу, то есть документы, которые действительно искал пользователь. Далее по списку релевантность информации в документах убывает и в конце оказываются менее всего подходящие документы к тому, что искал пользователь.

Одной из важных областей, где сегодня широко применяются данные методы, являются поисковые системы в интернете, где насчитывается огромное количество уникальных страниц. Используются такие методы и в различных организациях, где существует сравнительно высокий документооборот и зачастую поиск необходимой информации становится затруднительным. Также данные алгоритмы находят применения и в специализированных поисковых системах.

Цель данной работы исследовать и сравнить качества работы существующих методов, а также вариантов повышения качества их работы на выборке из поисковых запросов пользователей в интернете. В ходе работы рассматриваются два алгоритма для улучшения ранжирования. Первый алгоритм заключается в сглаживании рейтингов документов через релевантность к поисковому запросу соседних документов. Суть второго состоит в том, чтобы документам добавить признаки в виде принадлежности к некоторым тематикам и сравнивать с тематиками запроса.

В данной работе сначала будут описаны стандартные функционалы качества и рассмотрены типовые методы ранжирования для текстовых документов, так как текстовая информация является наиболее распространенная на сегодняшний день. Затем будут разобраны предложенные методы для улучшения качества ранжиро-

вания. И наконец, будут проведены исследования и сравнительный анализ методов ранжирования с предложенными улучшениями и без них на выборке из поисковых запросов.

2 Функционалы качества

Прежде чем описать основные меры качества, которыми будем пользоваться в экспериментах для анализа результатов, введем необходимые обозначения.

Пусть

q_i обозначает i поисковый запрос,

D_i – список всех документов, которые ассоциированы с q_i запросом,

$d_{i,j}$ – j-ый документ из списка D_i,

π_i — отсортированный список D_i,

$y_{i,j}$ – метка, показывающая на сколько документ $d_{i,j}$ релевантен к запросу q_i.

y_i – вектор меток $y_{i,j}$,

$[\cdot]$ – индикаторная функция.

Определим наиболее популярные и информативные функционалы качества алгоритмов ранжирования — NDCG и MAP меры качества.

2.1 NDCG функционал

NDCG(Normalized Discounted Cumulative Gain) мера качества является нормированной мерой от DCG(Discounted Cumulative Gain) меры, поэтому определим для начала DCG:

$$DCG@k(\pi_i, y_i) = \sum_{j:\pi_i(j)\leq k} G(y_{i,j})D(\pi_i(j)),$$

где k – количество первых документов в π_i, которые мы учитываем в функционале. Функция $G(y_{i,j})$ преобразовывает релевантность документа в его рейтинг, а $D(\pi_i(j))$ – функция значимости порядкового номера документа в ранжированном списке.

На практике часто функцию $G(y_{i,j})$ определяют как экспоненциальную, то есть $G(y_{i,j}) = 2^{y_{i,j}} - 1$, а $D(\pi_i(j))$ как единица, деленная на логарифм от номера позиции

документа: $D(\pi_i(j)) = \frac{1}{\log_2(1+\pi_i(j))}$. Таким образом, функционал намного слабее зависит от порядка документа в первых k позициях, но сильно зависит от релевантности документа.

В экспериментах использовалась стандартные значения функций. Из определения функций $G(y_{i,j})$ и $D(\pi_i(j))$ следует, что DCG мера качества имеет следующий вид:

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

NDCG функционал качества, как уже упоминалось, является нормированной величиной DCG так, чтобы результат находился на отрезке от нуля до единицы. Получаем, что для данной нормировки необходимо разделить $DCG@k(\pi_i, y_i)$ на максимальное значение, которое может быть у функционала качества при данных значениях π_i, y_i :

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

Популярность NDCG является следствием хороших свойств данной меры: она позволяет учитывать произвольным образом и порядок, и релевантность документов. Более того, данный функционал качества позволяет работать с разными значениями релевантности.

2.2 MAP функционал

В качестве второй удобной меры рассмотрим MAP(Mean Average Precision). Данный функционал учитывает уровень релевантности документа только как бинарная переменная: либо документ релевантный, либо – нет. Для определения MAP меры вначале определим функционалы точности P(Precision) и средней точности AP(Average Precision):

$$P@k(\pi_i, y_i) = \frac{\sum_{t:\pi_i(t)\leq\pi_i(k)}[y_{i,k} = 1]}{k},$$

$$AP@k((\pi_i, y_i) = \frac{\sum_{m=1}^{k} P@m(\pi_i, y_i)[y_{i,m} = 1]}{\sum_{m=1}^{k} y_{i,m}}.$$

Соответственно, $MAP@k$ определяется как среднее значение $AP@k$ по всем запросам q_i.Как видно, данный функционал также учитывает и порядок, и релевантность документов, что делает его таким же популярным.

Обе меры имеют параметр k – количество первых документов в отранжированном списке π_i, которые учитываются в функционалах. На практике k выбирается из постановки задачи или других априорных знаний.

3 Методы ранжирования

Существует большое количество методов ранжирования. Все методы условно можно классифицировать на три типа [5]: поточечные методы, которые независимо определяют рейтинг для каждого документа, попарные методы, идея которых в сравнении релевантности двух документов, то есть определить порядок для каждой пары документов, и списочные методы — моделирует порядок для всех документов в выдаче. С точки зрения поставленной задачи ранжирования, списочные методы являются методами, которые решают исходную постановку задачи.

Предложенная классификация строится из способа алгоритма обучения. На этапе прогнозирования и ранжирования методы имеют схожий алгоритм: для каждого документа $d_{i,j}$ вычисляется рейтинг $f(x_{i,j}, w)$, где $x_{i,j}$ — вектор признаков документ $d_{i,j}$, w — параметры метода ранжирования, далее рейтинги сортируются по убыванию. В итоге, получаем отранжированный список документов.

В задаче ранжирования признаки документов $x_{i,j}$ формируются не только из описания документов или только запроса, но и как функции от поискового запроса и документа: $x^{\ell}_{i,j} = g^{\ell}(q_i, d_{i,j})$, где $x^{\ell}_{i,j}$ – ℓ-признак документа $d_{i,j}$, а $g^{\ell}(q_i, d_{i,j})$ – некоторая функция. Такие признаки наиболее информативны, так как показывают степень релевантности документа к запросу. Рассмотрим наиболее часто используемые признаки после описания методов ранжирования.

В качестве алгоритмов ранжирования для сравнения выберем наиболее популярные методы из каждой категории: OC SVM (поточечный метод), Ranking SVM (попарный), LambdaRank(попарный), ListNet (списочный). Последовательно рассмотрем каждый метод.

3.1 OC SVM

OC SVM (One Class Support Vector Machine) является одним из базовых методов ранжирования [7]. Алгоритм заключается в построении параллельных гиперплоскостей, которые выступают в роли порогов. По взаиморасположение документа $d_{i,j}$ и гиперплоскостей (лежит он в положительном или отрицательном полупространстве каждой из гиперплоскости) делается предсказание о степени релевантности документа.

То есть пусть $y_{i,j} \in Y$, где $Y = \{1, 2, ..., \ell\}$, тогда метод строит гиперплоскости вида $\langle w, x_{i,j}\rangle - b_r = 0$, где r от 1 до $\ell - 1$, $w \in R^k$, $b_r \in R$ и $b_1 \leq b_2 \leq \cdots \leq b_{\ell-1} \leq b_\ell = +\infty$, чтобы правильно определять метки $y_{i,j}$ по следующей формуле:

$$y_{i,j} = \min_{r \in \{1,\ldots,\ell\}} \{r | \langle w, x_{i,j}\rangle - b_r < 0\}.$$

Перегруппируем все векторные представления документов $x_{i,j}$ по степени релевантности, то есть $\hat{x}_{r,k}$ — вектор признаков k-ого по счету документа, у которого степени релевантности равна r: $y_{r,k} = r$, и пусть таких документов m_r .Таким образом, становиться неважно, к какому запросу q_i относиться документ. Тогда задачу поиска таких w и b_r можно поставить следующим образом:

$$\frac{1}{2}||w||^2 + C\sum_{r=1}^{\ell-1}\sum_{k=1}^{m_r}(\xi_{r,k} + \hat{\xi}_{r+1,k}) \to \min_{w,b,\xi,\hat{\xi}},$$

$$\langle w, \hat{x}_{r,k}\rangle - b_r \leq -1 + \xi_{r,k},$$

$$\langle w, \hat{x}_{r,k}\rangle - b_r \geq 1 + \hat{\xi}_{r+1,k},$$

$$\xi_{r,k} \geq 0, \hat{\xi}_{r+1,k} \geq 0,$$

$$k = 1, \ldots, m_r, r = 1, \ldots . \ell - 1$$

Имеем стандартную задачу квадратичного программирования. Заметим, что алгоритм пытается проводить гиперплоскости, максимизируя отступы (расстояния) элементов до гиперплоскостей.

3.2 Ranking SVM

Данный метод был предложен Р. Хербрих [3] и имеет попарный подход. Ключевая идея алгоритма заключается в использовании алгоритма SVM для попарного сравнения элементов (документов) на то, какой из элементов более релевантный.

Пусть имеется два элемента x_i и x_j и некоторая функция $f(x)$, которая определяет релеватность объекта к запросу. Тогда, то что объект x_i релевантнее чем объект x_j ($x_i \succ x_j$) эквивалентно тому, что $f(x_i) > f(x_j)$:

$$x_i \succ x_j \iff f(x_i) > f(x_j).$$

Тогда, если выбрать функцию $f(x) = \langle w, x \rangle$, то имеем, что:

$$f(x_i) > f(x_j) \iff \langle w, x_i - x_j \rangle > 0.$$

И если теперь рассматривать разность векторных представлений как новые объекты $\hat{x}_{i,j} = x_i - x_j$, получим стандартную постановку SVM алгоритма. Формально задача Ranking SVM ставится следующим образом:

$$\begin{aligned}
&\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \to \min_{w,\xi},\\
&y_i\langle w, x_i^1 - x_i^2\rangle \geq 1 - \xi_i,\\
&\xi_i \geq 0,\\
&i = 1, \ldots, N,
\end{aligned}$$

Где N – количество пар объектов, x_i^1 – первый объект пары, x_i^2 – второй объект пары, а $y_i = 1$, если $x_i^1 \succ x_i^2$, и $y_i = -1$, если $x_i^1 \prec x_i^2$. Отметим, что пара из объектов x_i и x_j учитывается дважды – как пара (x_i, x_j) и (x_j, x_i) (соотвественно с $y_{i,j} = 1, y_{j,i} = -1$). В введенных ранее обозначениях допустимыми парами являются такие $x_{i,j}$ и $x_{r,k}$, что $i = r$ (то есть документы из одного списка) и $y_{i,j} \neq y_{i,k}$ (то есть имеют разную степень релевантности).

Данная постановка задачи позволяет использовать стандартную технику решения SVM – переход к двойственной задаче, и как следствие, делать нелинейные преобразования при помощи ядер SVM.

3.3 LambdaRank

В поточечных и попарных методах ранжирования итоговый функционал при обучении обычно не дифференцируемый, так как зависит от порядка элементов. В рассматриваемом алгоритме LambdaRank, описанный Бургес [2], вместо исходного функционала рассматривается непрерывный аналог, который легко оптимизировать. Вначале рассмотрим работу метода для одного поискового запроса.

Алгоритм LambdaRank не определяет непрерывный приближенный функционал L, вместо этого он определяет градиент функционала на всем списке документов:

$$\frac{\partial L}{\partial s_j} = -\lambda_j(s_1, y_1, \ldots, s_n, y_n),$$

Где $s_1, s_2, \ldots s_n$ – рейтинги документов, $y_1, y_2, \ldots, y_n$ – проставленные степени релевантности асессором, n – количество документов, найденных для поискового запроса. Таким образом, градиент по выбранному документу зависит от рейтингов и степени релевантности других документов. Знак выбран таким, чтобы положительное значение означало, что документ уменьшает функционал качества. Градиент для каждого документа пересчитывается после каждой генерации отранжированного списка. Функция градиента по всем документам называется лямбда функцией, отсюда и название метода.

Один из способов добиться повышения эффективности оптимизации — это увеличить градиенты документов на первых позициях, то есть более значимыми сделать перестановки для первых элементов. Пусть, например, имеются два релевантных документа d_1 и d_2 на позициях 2 и n в отсортированном списке. Тогда перестановка документа d_1 на первую позицию должно требовать меньших затрат, чем перестановка документа d_2 на первую позицию, то есть градиент у d_1 должен быть значительно больше. В общем случае, для двух документов d_i и d_j имеем, если $d_i \succ d_j$, то

$$|\frac{\partial L}{\partial s_i}| > |\frac{\partial L}{\partial s_j}|.$$

Метод позволяет настраиваться на широкий класс функционалов качеств, однако в стандартном варианте (который и будет использоваться) λ вычисляется по функционалу NDCG:

$$\lambda_j = \frac{\partial L}{\partial s_j} = \frac{1}{G_{\max}} \sum_i \Big(\frac{1}{1+\exp(s_i - s_j)}\Big)(G(y_i) - G(y_j))(D(\pi_i) - D(\pi_j)),$$

где функции $G(\cdot), D(\cdot)$ — функции преобразования релевантности документа в его рейтинг и значимости порядкового номера документа в ранжированном списке (из определения NGCG меры). λ_j – показывает насколько надо увеличить рейтинг j документа. Для этого надо изменить веса w (напомним, что рейтинг высчитывается как: $s_i = \langle w, x_{r,i} \rangle$):

$$\frac{\partial L}{\partial w} = \sum_{(i,j) \in P} \Big(\frac{\partial L(s_i, s_j)}{\partial s_i} \frac{\partial s_i}{\partial w} + \frac{\partial L(s_i, s_j)}{\partial s_j} \frac{\partial s_j}{\partial w} \Big),$$

где P — множество пар документов. Пусть $P_{i\_}$ — множество документов j, для которых пары документов (i, j) в множестве P, а $P_{\_j}$ — множество документов i для которых пары документов (i, j) в множестве P, тогда $\frac{\partial L}{\partial w}$ можно записать как:

$$\frac{\partial L}{\partial w} = \sum_{i=1}^{n} \frac{\partial s_i}{\partial w} \sum_{j \in P_{i\_}} \frac{\partial L(s_i, s_j)}{\partial s_i} + \sum_{j=1}^{n} \frac{\partial s_j}{\partial w} \sum_{i \in P_{\_j}} \frac{\partial L(s_i, s_j)}{\partial s_j},$$

Тем самым понизив вычислительную сложность итерации с $O(n^2)$ до $O(n)$. Таким образом, алгоритм LambdaRank заключается в итерационном пересчете весов w:

$$w = w - \eta \frac{\partial L}{\partial w},$$

где η — итерационный шаг. Очевидно, что если запросов несколько, то все изменения заключаются в расширении множества P – допустимых пар, которое может быть построено как и в методе Ranking SVM. Отметим, что приближенный непрерывный функционал L определяется на списке документов, что позволяет рассматривать LambdaRank не только как попарный метод, но и как списочный.

3.4 ListNet

Метод ListNet имеет списочный подход и основан на вероятностной модели Plackett-Luce. Впервые алгоритм был описан З. Као [4]. Прежде чем перейти к методу опишем модель Плакетт-Люис.

Пусть дан отранжированный список π, и рейтинг документов $s = \{s_1, s_2, \ldots, s_n\}$. Обозначим за $\pi^{-1}(i)$ документ в позиции i отранжированного списка π. Тогда

Plackett-Luce модель (кратко PL-модель) задает вероятность порядка π при помощи рейтингов s как плотность вероятности следующим образом:

$$P_s(\pi) = \frac{s_{\pi^{-1}(1)}}{\sum_{j=1}^{n} s_{\pi^{-1}(j)}} \frac{s_{\pi^{-1}(2)}}{\sum_{j=2}^{n} s_{\pi^{-1}(j)}} \cdots \frac{s_{\pi^{-1}(n)}}{s_{\pi^{-1}(n)}} = \prod_{i=1}^{n} \frac{s_{\pi^{-1}(i)}}{\sum_{j=i}^{n} s_{\pi^{-1}(j)}}.$$

Данное распределение имеет несколько полезных свойств. Во-первых, список документов с убывающим рейтингом имеет наибольшую вероятность по сравнению со всеми другими перестановками в списке, а с возрастающим рейтингом — наименьшую вероятность. Как следствие, перестановка двух документов в убывающем по рейтингу списке приведет к уменьшению вероятности.

Во-вторых, PL-модель можно определить на некотором подмножестве первых документов в списке и по-прежнему это будет функцией распределения. Пусть хотим рассмотреть только k первых документов в списке, тогда:

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

Более того, верно следующее утверждение:

$$P_s^k = \sum_{\pi \in Q} P_s(\pi),$$

где Q — множество перестановок документов, где первых k документов те же, что и в убывающем по рейтингу списке документов.

Алгоритм ListNet может быть применен с PL-распределением по всем документам в списке. Однако, в данном случае вычислительная сложность метода становиться слишком большой — $O(n!)$. Поэтому используют PL-распределение на первых k документах ($O(\frac{n!}{(n-k!)})$).

В нотации PL-модели значения релевантности первых k документов из списка можно задать следующим образом:

$$P_y^k = \prod_{i=1}^{k} \frac{exp(y_i)}{\sum_{j=i}^{n} exp(y_j)}.$$

Тогда результаты работы метода ListNet в той же форме можно записать как:

$$P_{F(x,w)}^k = \prod_{i=1}^{k} \frac{exp(f(x_i, w))}{\sum_{j=i}^{n} exp(f(x_j, w))},$$

где $f(x, w)$ — модель нейронной сети с параметрами w, а $F(x, w)$ — список рейтингов, выданные моделью.

Если P_y^k и $P_{F(x,w)}^k$ имеют почти схожие значения, то рейтинги документов, полученные алгоритмом, будут иметь приблизительно те же значения, что и проставленные асессором значения релевантности. Таким образом, достаточно $P_{F(x,w)}^k$ приближать к P_y^k. Для измерения разницы между распределениями и последующей оптимизации удобно использовать KL-дивергенцию. В этом и состоит идея алгоритма. Напомним, что KL-дивергенция в дискретном случае для распределений P и Q имеет следующий вид:

$$KL(P||Q) = \sum_i p_i \log \frac{p_i}{q_i} = \sum_i p_i \log p_i - \sum_i p_i \log q_i,$$

и равна нулю тогда и только тогда, когда распределения P и Q совпадают. Получаем, что задача для минимизации функции потерь имеет следующий вид:

$$L(w) = \sum_{i=1}^{m} L_i(y^i, F(x^i, w)) \to \min_w,$$

$$L_i(y^i, F(x^i, w)) = KL(P_{y^i}^k || P_{F(x^i,w)}^k) = -\sum_{g \in G_i^k} P_{y^i}^k(g) \log P_{F(x^i,w)}^k(g)$$

где L_i — функция потерь для i поискового запроса, m — количество запросов, G_i^k — множество всевозможных значений для первых k элементов списка. Заметим, что в формуле опущен первый член KL-дивергенции, так как он не зависит от w. Градиент функции потери можно найти по следующей формуле:

$$\frac{\partial L(w)}{\partial w} = \sum_{i=1}^{m} \frac{\partial L_i(y^i, F(x^i, w))}{\partial w},$$

$$\frac{\partial L_i(y^i, F(x^i, w))}{\partial w} = -\sum_{g \in G_i^k} \frac{P_{y^i}^k(g)}{P_{F(x^i,w)}^k(g)} \frac{\partial P_{F(x^i,w)}^k(g)}{\partial w}.$$

Метод ListNet заключается в итерационном пересчете $\frac{\partial L(w)}{\partial w}$ и обновлении весов модели: $w = w - \eta \frac{\partial L(w)}{\partial w}$. В экспериментах алгоритм использовался при $k = 10$ и с одним полносвязным скрытым слоем нейронной сети.

4 Стандартные признаки для ранжирования

Теперь рассмотрим, как именно формируются признаки из пары запрос-документ. В общем случае, постановка задачи ранжирования подразумевает наличие запроса q_i, который представляет из себя последовательный набор слов, и списка документов D_i, которые в свою очередь можно представить как последовательный наборы слов в названии и тексте документов.

Если детальнее рассматривать поисковую выдачу web-страниц, то в качестве признаков также используются популярность web-ресурса, цитируемость, количество страниц, на которые ссылается страница и другие признаки [6] . Однако, для обобщения остановимся только на текстовых представлениях документа и запроса.

Введем некоторые дополнительные обозначения. Пусть

T — множество всех употребляемых терминов (слов или словосочетаний),

$N = |T|$ — количество всех терминов,

q_i^k — количество вхождение k-термина в q_i поисковый запрос,

$q_i = (q_i^1, q_i^2, \ldots, q_i^N)$ — векторное представление q_i запроса,

$t_{i,j}^k$ — количество вождений k-термина в текст $d_{i,j}$ документа,

$d_{i,j}^b = (t_{i,j}^1, t_{i,j}^2, \ldots, t_{i,j}^N)$ — векторное представление текста $d_{i,j}$ документа,

$\hat{t}_{i,j}^k$ — количество вхождений k-термина в название $d_{i,j}$ документа,

$d_{i,j}^t = (\hat{t}_{i,j}^1, \hat{t}_{i,j}^2, \ldots, \hat{t}_{i,j}^N)$ – векторное представление названия $d_{i,j}$ документа,

$d_{i,j} = (d_{i,j}^b, d_{i,j}^t)$ — векторное представление $d_{i,j}$ документа.

В данной работе, как часто и делают на практике, в качестве терминов выбраны слова. Такие векторные представления описывают поисковый запрос и список документов для ранжирования. Однако, для обучающихся алгоритмов признаки в таком формате мало информативны, так как не показывают взаимосвязи между запросом и документом. Прежде чем перейти к конечным признакам, опишем полезные преобразования: TF(term frequency), IDF(inverse document frequency) и TF-IDF.

TF-преобразование — это отношение числа вхождений термина к общему числу слов в документе:

$$tf(t^k, d_{i,j}) = \frac{t_{i,j}^k}{\sum_{\ell=1}^{N} t_{i,j}^\ell},$$

IDF-преобразование — это логарифм отношения общего числа документов к документам, где встречается данный термин:

$$idf(t^k) = \log \frac{|D|}{\sum_{t=1}^{T} \sum_{\ell=1}^{D_i} [t_{t,\ell}^k > 0]},$$

где D – все документы ($|D|$ — их общее количество), T – количество поисковых запросов. Как видно, IDF-преобразование определяется для термина, когда TF-преобразование определяется для пары термин-документ. TF-IDF является произведением TF и IDF преобразований:

$$tfidf(t^k, d_{i,j}) = tf(t^k, d_{i,j}) \cdot idf(t^k).$$

Наконец, опишем признаки для пары $(q_i, d_{i,j})$, которые используются для обучения алгоритмов:

$\sum_{t^k \in q_i \cap d_{i,j}^b} tf(t^k, d_{i,j}^b)$	сумма TF-частот слов из запроса в тексте
$\sum_{t^k \in q_i \cap d_{i,j}^t} tf(t^k, d_{i,j}^t)$	сумма TF-частот слов из запроса в названии
$\sum_{t^k \in q_i \cap d_{i,j}} tf(t^k, d_{i,j})$	сумма TF-частот слов из запроса во всем документе
$\sum_{t^k \in q_i \cap d_{i,j}^b} idf(t^k)$	сумма IDF-частот слов, встречаемых в запросе и тексте
$\sum_{t^k \in q_i \cap d_{i,j}^t} idf(t^k)$	сумма IDF-частот слов, встречаемых в запросе и названии
$\sum_{t^k \in q_i \cap d_{i,j}} idf(t^k)$	сумма IDF-частот слов, встречаемых в запросе и документе
$\sum_{t^k \in q_i \cap d_{i,j}^b} tfidf(t^k, d_{i,j})$	сумма TF-IDF-частот слов из запроса в тексте
$\sum_{t^k \in q_i \cap d_{i,j}^t} tfidf(t^k, d_{i,j})$	сумма TF-IDF-частот слов из запроса в названии
$\sum_{t^k \in q_i \cap d_{i,j}} tfidf(t^k, d_{i,j})$	сумма TF-IDF-частот слов из запроса во всем документе
$\sum_{k=1}^{N} t_{i,j}^k$	длина текста
$\sum_{k=1}^{N} \hat{t}_{i,j}^k$	длина названия
$\sum_{k=1}^{N} (\hat{t}_{i,j}^k + t_{i,j}^k)$	длина всего документа

К данным признакам часто добавляют признаки, которые являются результатами простейших моделей ранжирования, такие как BM25 и LMIR [6]. Поскольку обе модели требует настройки некоторых параметров, то для чистоты экспериментов не будем их использовать.

Как видно, в таком формате признаки представляют собой некоторые частотные оценки встречаемости слов во всем документе и отдельных его частях. При таком количестве признаков и их смысловой нагрузки не так много пространства для обучения алгоритмов. Можно в качестве признаков использовать частотные оценки по каждому термину. Однако, в этом случае не совсем понятно, как применять алгоритмы, потому что теряется возможность ранжировать списки для новых запросов, которые не были в обучающей выборке, что является одним из важнейших требований.

Рассмотрим несколько вариантов, как можно улучшить результаты ранжирования.

5 Сглаживание рейтингов документов

На практике в качестве поисковых запросов часто выступает короткий набор слов. Как показывалось выше, признаки будут состоять только из частотных оценок встречаемости этих слов в документах. При этом, если документ является релевантным и даже содержит синонимичные слова, то он будет проигнорирован, поскольку нет слов из короткого поискового запроса.

То есть проблема заключается в том, что запрос слишком короткий, чтобы можно было оценить некоторые документы "по-настоящему" на сколько релевантен он запросу. Пусть справедливо следующее предположение: размер документов в среднем значительно больше, чем размер поискового запроса. Тогда если два документы похожи по смыслу, то можно оценить релевантность одного документа через релевантность второго.

Рассмотрим пример. Пусть есть запрос q, состоящий из двух слов q^1 и q^2, и три документа d_1, d_2, d_3, каждый из которых состоит из тысячи слов, и первых два документа релевантны к запросу. И пусть у d_1 документа слова q^1 и q^2 встречаются, а у остальных нет, и при этом остальные слова у d_1 такие же как и у d_2. Очевидно, что требуется, чтобы алгоритм отражнировал документы по порядку, то есть $d_1 \succ d_2 \succ d_3$ (или $d_2 \succ d_1 \succ d_3$). Однако, с точки зрения алгоритма, не будет никакой разницы между порядками $d_1 \succ d_2 \succ d_3$ и $d_1 \succ d_3 \succ d_2$, так как документы d_2, d_3

будут иметь идентичное признаковое представление. Но если мы учтем, что первые два документа очень сильно похожи и частично учтем рейтинг d_1 документа для d_2, то получим в точности правильный порядок.

Опишем метод сглаживания рейтингов более формально. Пусть алгоритм ранжирования для поискового запроса q_i и списка документа выдал рейтинги документов $f(x_{i,1}, w), f(x_{i,2}, w), \ldots, f(x_{i,n_i}, w)$, тогда конечный рейтинг $s_{i,j}$ документа $d_{i,j}$ будем вычислять следующим образом:

$$s_{i,j} = f(x_{i,j}, w) + \alpha\big(\sum_{k=1}^{K} a_{j,z_k} f(x_{i,z_k}, w)\big),$$

где α и K — параметры, показывающие насколько сильно учитывать рейтинги других документов и сколько наиболее близких по смыслу документов рассматривать, a_{j,z_k} — степень близости между $d_{i,j}$ и d_{i,z_k} документами, а z_k — k-ый по близости документ $d_{i,j}$.

В качестве меры близости для документов наиболее информативна косинусная мера. То есть a_{j,z_k} высчитываются по формуле:

$$a_{j,z_k} = \frac{\langle d_{i,j}, d_{i,z_k} \rangle}{||d_{i,j}||_2 ||d_{i,z_k}||_2},$$

где $||\cdot||_2$ — евклидова норма. Косинусная мера для пары документов равна $\rho(d_{i,j}, d_{i,z_k}) = 1 - a_{j,z_k}$.

Отметим, что метод не вносит никаких изменений в алгоритм ранжирования. До начала формирования признаков для пары запрос-документ, надо для каждого документа найти похожие по косинусной мере документы из списка выдачи. И после работы алгоритма учесть в рейтинге документа рейтинги схожих документов. Для данного метода можно провести аналогию с алгоритмом k-ближайших соседей. Тем самым можно улучшить работу метода, используя разные виды алгоритмов из семейства методов k-ближайших соседей.

Вообще говоря, чтобы найти несколько ближайших документов надо перебрать все документы, то есть сложность алгоритма $O(|T|n^2)$, где $|T|$ — множество всех слов в словаре, а n — количество документов в выданном списке. Такая сложность может быть затруднительной в момент ранжирования списка. Но так как документы поступают не в момент запроса, а уже загружены в систему ранжирования, и

мера близости не зависит от запроса, то близкие по смыслу документы могут быть посчитаны заранее.

6 Добавление тематики

Основная проблема стандартных используемых признаков заключается в том, что одни и те же вещи можно описать несколькими разными словами, при этом смысл будет одним и тем же. Но так как алгоритм не может определить по словам, что смысл один и тот же, то документы, где нет слов из с поискового запроса, будут проигнорированы.

Попробуем решить эту проблему другим способом. Суть задачи ранжирования состоит в поиске документов, подходящих по смыслу к поисковому запроса. Попытаемся определить этот смысл запроса и смысл документов, и сопоставить, насколько эти смыслы похожи. Достоверно смысл документа и запроса определять не требуется, надо только определить схожесть смысла документа и запроса. Здесь неявно используется тот факт, что и запрос и документы представляются словами, то есть из одного признакового пространства.

Из выше сказанного можно сделать вывод, что достаточно будет определить насколько сильно принадлежит документ и запрос к некоторым темам. То есть получить вектор принадлежности к некоторым темам как запросов, так и документов. Далее остается только сравнить вектора запроса и документов, где опять можно воспользоваться косинусной мерой. Таким образом, получаем дополнительный признак — схожесть тем. Так как схожесть тематик является достаточно информативным признаком, то можно использовать и несколько мер или рассматривать разность вероятностей принадлежностей к темам поискового запроса и документа. Однако, для упрощения экспериментов будем использовать только косинусную меру.

Рассмотрим алгоритм подробнее. Для моделирования тематик воспользуемся моделью латентного размещения Дирихле (LDA) [1]. Данная модель позволяет разбить список документов на произвольное количество тематик, причем один документ может принадлежать сразу к нескольким темам. Каждая тематика рассматривается

как некоторое распределение вероятностей в пространстве слов из общего словаря. Вероятностная модель LDA задается как

$$p(T, Z, \Theta|\Phi, \alpha) = \prod_{d=1}^{D} p(\theta_d|\alpha) \prod_{n=1}^{N_d} p(t_{d,n}|z_{d,n}, \Phi)p(z_{d,n}|\theta_d),$$
$$p(\theta|\alpha) = Dir(\theta_d|\alpha),$$
$$p(t_{d,n}|z_{d,n}, \Phi) = \Phi_{z_{d,n},t_{d,n}},$$
$$p(z_{d,n}|\theta_d) = \Theta_{d,z_{d,n}},$$

Где $Dir(\cdot|\cdot)$ — распределение Дирихле, θ_d – вероятности тематик в документе d, $z_{d,n}$ – тематика n-ого слова в документе d, Φ – вероятности встретить термин в каждой тематике ($\Phi \in R^{C\times T}$, где C – количество тематик), а Z и Θ – разбиение всех слов по тематикам и вероятности тематик во всех документах соответственно. И требуется найти максимум распределения $p(T|\Phi, \alpha)$ по Φ, α.

Так как в экспериментах используется реализованный метод LDA, то не будем здесь вдаваться в подробности алгоритма обучения модели. Отметим, что метод можно обучать на множестве документов, который не имеет отношений к выборке, так как не ставиться задача разбить документы выборки по темам. Что позволяет обучать алгоритм на большем множестве документов и точнее определять принадлежность слов к некоторым темам. Однако, нельзя забывать, что другой набор данных может сильно не походить на типовые документы из задачи ранжирования, что может привести к ухудшению определения тематик. В данной работе алгоритм LDA обучался на документах из поисковой выдачи документов.

7 Эксперименты

Проведем сравнительные эксперименты по работе алгоритмов ранжирования, а также сравним работу методов при использовании сглаживания рейтингов, добавлении меры схожести тематик и использовании сглаживания рейтингов с добавлением меры схожести тематик.

В качестве выборки используется данные о 915 поисковых запросах, где для каждого запроса проставлена метка из трех типов релевантности: 0 – документ не ре-

левантен, 1 – документ релевантен, 2 – документ сильно релевантен. Для каждого поискового запроса в выдаче размечено от 30 до 50 документов в зависимости от запроса, причем около 30-40% из них имеют положительную метку. Каждый документ является вебстраницей и представляет собой словарь из шести полей: url – адрес страницы, title – название страницы, meta – мета-данные, текст указанный в некоторых тегах вебстраницы для поисковых систем, menu – текст страницы, относящийся к навигации (меню) по сайту, footer – контактная информация администраторов сайта, а также дополнительная навигация, body – остальной текст на вебстранице. Поисковый запрос представляет собой набор слов.

В качестве признаков для пары запрос-документ используется сумма TF, IDF и TF-IDF-частот слов, встречаемых в запросе и частей документа title, menu, body, footer, а так же и всего документа. Также используются длины текста во всех полях документа (в том числе url).

Так как выборка не слишком большая, то для чистоты экспериментов, будем делать кросс-валидацию по 5 подвыборкам, которые состоят из 183 поисковых запросов и получены случайном образом из генеральной совокупности. Каждый раз для обучения будем использовать 4 подвыборки и одну для контроля, а результаты по пяти конфигурациям усреднять.

Для начала сравним работу методов между собой.

7.1 Сравнение работы алгоритмов ранжирования

Сравнивать работу алгоритмов будем сразу для MAP и NDCG функционалов качеств при $k = 1, 5, 10$. Также в таблицу для сравнения включим случайным образом сгенерированный порядок документов в поисковой выдачи. Это нужно, чтобы сравнить, как работают алгоритмы в целом, так как даже при случайном выборе функционалы качества не стремятся к нуля (это связано с тем, что немалая часть документов в выдаче имеют положительный результат и при случайном выборе так же берутся документы и с положительной меткой).

Результаты экспериментов в таблице 1. Как видно, попарные и списочные методы значительно лучше работают, чем поточечный метод, и почти в два раза лучше, чем

Algorithm	MAP@1	MAP@5	MAP@10	NDCG@1	NDCG@5	NDCG@10
Random	0.388	0.293	0.281	0.267	0.294	0.353
OC SVM	0.637	0.489	0.401	0.476	0.414	0.475
RankingSVM	0.710	0.564	0.517	0.567	0.542	0.583
LambdaRank	0.718	0.574	**0.526**	0.569	**0.560**	**0.600**
ListNet	**0.721**	**0.576**	0.521	**0.572**	0.553	0.598

Таблица 1: Сравнение работы алгоритмов

случайный выбор документов. Напомним, что данные имеют три степени релевантности, что в частности отображает тот факт, что $NDCG@1 \neq MAP@1$.

LambdaRank и ListNet примерно одинаково работают. Отметим, что LambdaRank при обучении настраивается на NDCG функционал качества.

7.2 Сглаживание рейтингов документов

Рассмотрим работу метода при разном количестве соседних документов K, рейтинги которых учитываются, и разном значении параметра α — параметре, показывающем, насколько сильно необходимо учитывать рейтинги соседних документов.

Для каждого значения K наилучшим значением параметра α было в интервале от 5 до 10. Для наглядности на рисунке 1 приведены результаты экспериментов для наиболее показательных значений параметров по мере MAP@5. Остальные рассматриваемые функционалы ведут себя идентичным образом на данных.

Лучшее значение для функционала MAP@5 достигается при $K = 8$ и $\alpha = 10$, причем улучшение составляет больше чем 15% . Отметим, что при $\alpha > 10$ наблюдается некоторая стабилизация, что является следствием того, что собственный рейтинг учитывается значительно меньше и конечный рейтинг строится по близким по смыслу документам.

Такая оптимальность по функционалу при сравнительно больших значениях K объясняется тем, что рейтинги соседних документов берутся с коэффициентами "близости"a_{j,z_k}, поэтому если ближайшие по косинусной мере документы находятся далеко, они слабо учитываются.

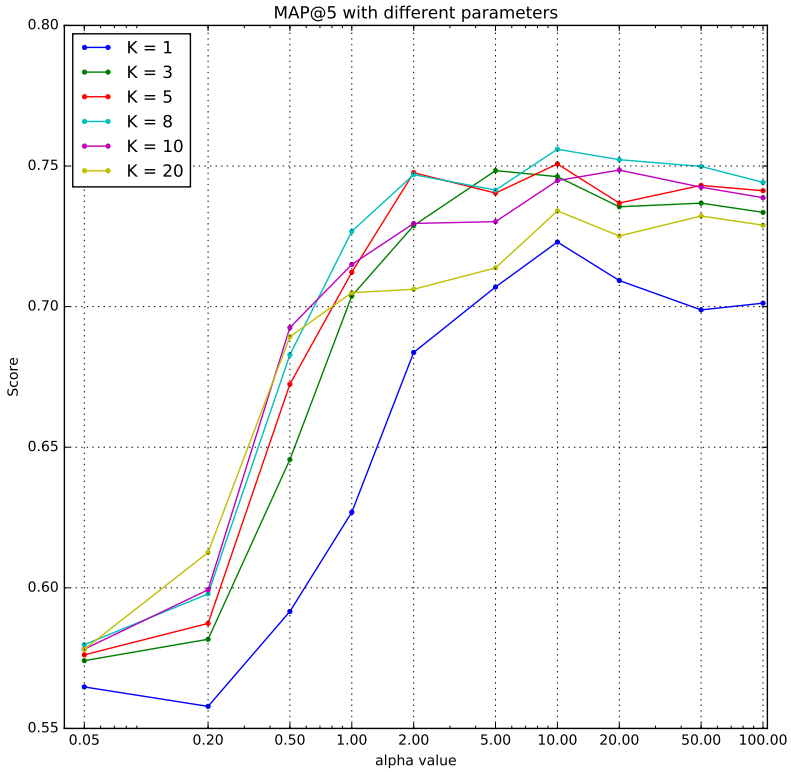

Рис. 1: Значение функционала MAP@5 при разных параметрах

Результаты экспериментов по всем алгоритмам и рассматриваемым функционалов приведены в таблице 2. В таблице указаны лучшие значения при разных параметрах K, α. Для удобного сравнения в первой графе ('Best without') приведены лучшие результаты из таблицы 1.

Заметим, что функционалы MAP@k повысились больше чем на 16%, а NDCG@k повысились в среднем на 11%. Вероятная причина данной разницы заключается в том, что документы с релевантностью 1 и 2 с точки зрения алгоритма сглаживания неразличимы, то есть разница между косинусными мерами документов с релевантностями 2 и 1 и 2 и 2 не слишком большая. А так как документов с релевантностями 2 меньше чем с 1, то документы с релевантностью 2 больше настраиваются на документы с релевантностью 1. Таким образом, в начале отранжированного списка получается больше документов с релевантностью 1.

Algorithm	MAP@1	MAP@5	MAP@10	NDCG@1	NDCG@5	NDCG@10
Best without	0.721	0.576	0.526	0.572	0.560	0.600
OC SVM	0.713	0.602	0.585	0.576	0.534	0.561
RankingSVM	0.823	0.737	0.714	0.688	0.670	0.698
LambdaRank	**0.841**	**0.754**	**0.725**	0.696	**0.682**	**0.716**
ListNet	0.839	0.752	0.719	**0.699**	0.678	0.703

Таблица 2: Сравнение работы алгоритмов для сглаживания рейтингов

7.3 Сравнение работы алгоритмов при добавлении тематик

Вообще говоря, у алгоритма LDA есть несколько параметров, которые надо настраивать под данные. Однако, так как не требуется точно приблизить разбиение тематик по документам, а лишь получить векторное представление по документам, чтобы сравнить с тематиками поискового запроса, то для настраивания выберем только количество тематик.

Рассмотрим работу алгоритмов при $N = \{10, 25, 50, 75, 100, 150, 200\}$ темах. На рисунках 2 - 3 приведены результаты для функционалов MAP@5 и NDCG@10. Как видно, лучшие результаты достигаются при $N = 100$.

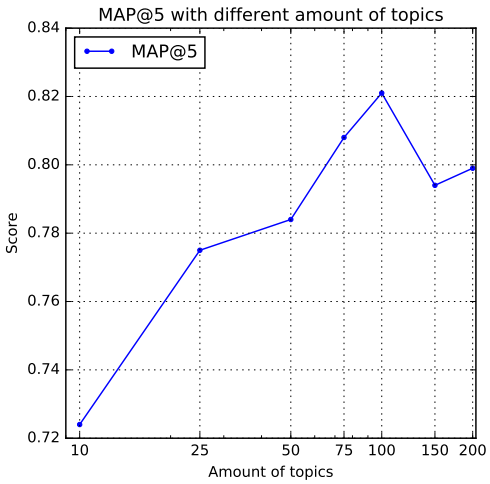

Рис. 2: MAP@5 при разном количестве тематик

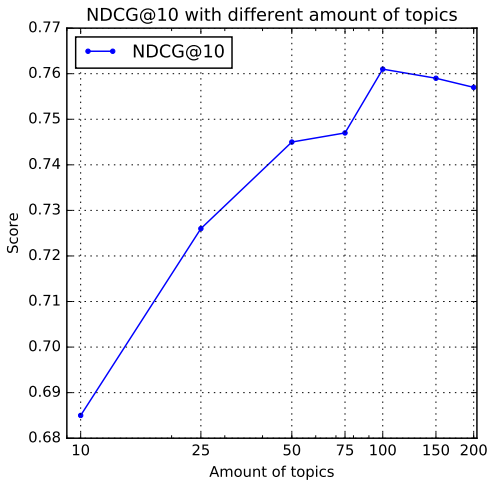

Рис. 3: NDCG10 при разном количестве тематик

Algorithm	MAP@1	MAP@5	MAP@10	NDCG@1	NDCG@5	NDCG@10
Best without	0.721	0.576	0.526	0.572	0.560	0.600
OC SVM	0.749	0.679	0.641	0.574	0.606	0.623
RankingSVM	0.840	0.817	0.755	0.685	0.726	0.753
LambdaRank	**0.851**	**0.820**	**0.767**	**0.697**	0.738	**0.761**
ListNet	0.843	**0.820**	0.764	0.695	**0.739**	0.755

Таблица 3: Сравнение работы алгоритмов при добавлении признаков

Algorithm	MAP@1	MAP@5	MAP@10	NDCG@1	NDCG@5	NDCG@10
Best without	0.721	0.576	0.526	0.572	0.560	0.600
OC SVM	0.763	0.703	0.672	0.597	0.641	0.656
RankingSVM	0.853	0.822	0.764	0.690	0.735	0.766
LambdaRank	**0.863**	**0.828**	**0.771**	**0.701**	**0.747**	**0.775**
ListNet	0.851	0.825	0.770	0.698	0.744	0.769

Таблица 4: Сравнение работы алгоритмов при добавлении признаков и сглаживании рейтингов

Лучшие результаты экспериментов приведены в таблице 3. Заметим, что добавление тематик сильно улучшило показатели функционалов, которые учитывают первых 5 и 10 объектов в ранжированном списке. В среднем наблюдается улучшение на 19% и на 15% по MAP и NDCG мерам соотвественно.

7.4 Сравнение работы алгоритмов при добавлении тематик и сглаживании рейтингов

Рассмотрим улучшение при работе обоих методов. Параметры для алгоритмов возьмем те же самые. В таблице 4 приведены лучшие результаты.

Из результатов экспериментов следует, что вместе алгоритмы не слишком сильно улучшают качество функционалов. Однако, в итоге получается улучшить функционалы качества в среднем на 18%.

8 Заключение

В данной работе были рассмотрены несколько методов ранжирования и проведен их сравнительный анализ на выборке из поисковых запросов. Также были предложены и рассмотрены алгоритмы сглаживания рейтингов и добавление тематических признаков. Для проведения экспериментов были реализованы описанные методы и алгоритмы на языке Python.

В ходе данной работы были получены следующие результаты:

- Среди рассматриваемых алгоритмов ранжирования OC SVM, Ranking SVM, LambdaRank, ListNet лучшие показатели у LambdaRank,
- Алгоритм сглаживания рейтингов через рейтинги соседних документов улучшает работу в среднем на 13%,
- Добавление тематических признаков улучшает работ в среднем на 17%,
- При применении обоих методов получается улучшить функционалы качества в среднем на 18% .

Таким образом, были решены и рассмотрены все поставленные задачи: исследовать и сравнить качества работы существующих методов ранжирования из разных классов, предложить алгоритмы для улучшения качества работы методов ранжирования на выборке из поисковых запросов.

Список литературы

[1] *Blei D. M., Ng A. Y., Jordan M. I.* Latent dirichlet allocation // *J. Mach. Learn. Res.* — 2003. — Vol. 3. — Pp. 993–1022.

[2] *Burges C. J., Ragno R., Le Q. V.* Learning to rank with nonsmooth cost functions // Advances in Neural Information Processing Systems 19 / Ed. by P. B. Schölkopf, J. C. Platt, T. Hoffman. — MIT Press, 2007. — Pp. 193–200.

[3] *Herbrich R., Graepel T., Obermayer K.* Support vector learning for ordinal regression // In International Conference on Artificial Neural Networks. — 1999. — Pp. 97–102.

[4] Learning to rank: From pairwise approach to listwise approach / Z. Cao, T. Qin, T.-Y. Liu et al. // Proceedings of the 24th International Conference on Machine Learning. — ICML '07. — New York, NY, USA: ACM, 2007. — Pp. 129–136.

[5] *Li H.* Learning to Rank for Information Retrieval and Natural Language Processing. — Morgan & Claypool Publishers, 2011.

[6] *Liu T.-Y.* Learning to rank for information retrieval // *Found. Trends Inf. Retr.* — 2009. — Vol. 3, no. 3. — Pp. 225–331.

[7] *Shashua A., Levin A.* Ranking with large margin principle: Two approaches // In Proceedings of Advances in Neural Information Processing Systems / Ed. by NIPS. — 2003.