

Текстовое ранжирование II

Информационный поиск. Лекция №6



Андрей Кривой

О преподавателях



Андрей Кривой

Руководитель группы ранжирования.

Занимаюсь качеством поиска в поисковых проектах VK

Telegram: @mt6qmzaotzn3

Лекции

Часть I: инженерная

1. Введение
2. Поисковый робот
3. Индексация и булев поиск
4. Предобработка запросов. Исправление опечаток

Часть III: нейропоиск

9. Нейропоиск: семантический матчинг
10. Нейропоиск: переранжирование
11. Нейропоиск: векторный поиск
12. Мультимедийный поиск

Часть II: ранжирование

5. Текстовое ранжирование I: метрики и векторные модели
6. Текстовое ранжирование II: вероятностные модели
7. Поведенческое и ссылочное ранжирование
8. Машинное обучение ранжированию

Сегодня мы тут!



О чем будем говорить

1. Метрики качества поиска
2. Вероятностные модели, BM25
3. Расширения текстового ранжирования (зоны, порядок и близость слов, ...)
4. Семинар и ДЗ

TF-IDF



Вспоминаем TF-IDF

- Формула TF-IDF – простейший способ ранжировать документы
- TF-IDF можно понимать как скалярное произведение:
 $TfIdfScore(q, d) = \mathbf{q} * \mathbf{d}$
- в котором запрос и документ представлены в виде векторов \mathbf{q} и \mathbf{d} размером M , где M – полное число известных терминов (размер словаря):
 - $\mathbf{q} = (0, 1, 0, 0, 1, \dots, 0, 1)$ где компонент q_i на i -й позиции равен 1 если i -й термин v_i из словаря присутствует в запросе, и 0 – если нет
 - $\mathbf{d} = (\dots, TF(v_i, d) * IDF(v_i), \dots)$ где компонент d_i на i -й позиции равен TF-IDF i -го термина v_i из словаря (и, очевидно, он 0 если такого термина нет в документе)

$$TfIdfScore(q, d) = \sum_{t_i \in q} TF(t_i, d) * IDF(t_i)$$

тут:

- $TF(t_i, d)$ – сколько раз термин t_i встретился в документе d (т.н. *term frequency*)
- $IDF(t) = \log \frac{N}{DF(t_i) + 1} + 1$ – мера «глобальной» важности термина t_i в корпусе документов (т.н. *inverted document frequency*)
- N – полное число документов в корпусе
- $DF(t_i)$ – число документов в корпусе, содержащих термин t_i

Модель векторного пространства (VSM)

- TF-IDF можно обобщить до идеи векторного пространства (*Vector Space Model*, или просто *VSM*)
- Представляем запросы и документы в виде (очень!) больших разреженных векторов размером словаря
- Ранжируем документы по скалярному произведению (или косинусу) между векторами запроса и документа
- В качестве компонентов наших векторов можно использовать TF, IDF, TF*IDF и их всевозможные варианты, например:
 - можно использовать логарифмы от TF
 - векторы запросов и документов можно нормализовать
 - и т.д.
- В примерах из прошлого семинара мы представляли и документы, и запросы в виде нормализованных векторов TF-IDF т.к. это дефолтное поведение *TfidfVectorizer*'а из библиотеки *scikit-learn*

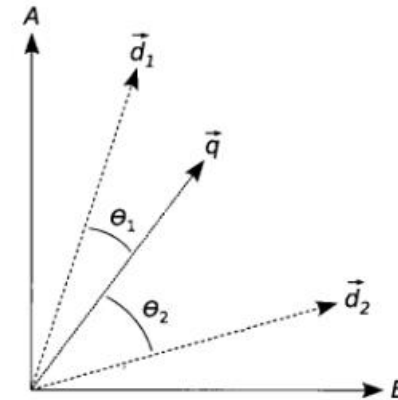


Figure 2.8 Document similarity under the vector space model. Angles are computed between a query vector \vec{q} and two document vectors \vec{d}_1 and \vec{d}_2 . Because $\theta_1 < \theta_2$, d_1 should be ranked higher than d_2 .

Иллюстрация идеи векторного пространства из книги Butcher'a, ищем косинусное расстояние между запросом **q** и документами **d1** и **d2**.

Метрики качества поиска





You can't improve what you don't
measure

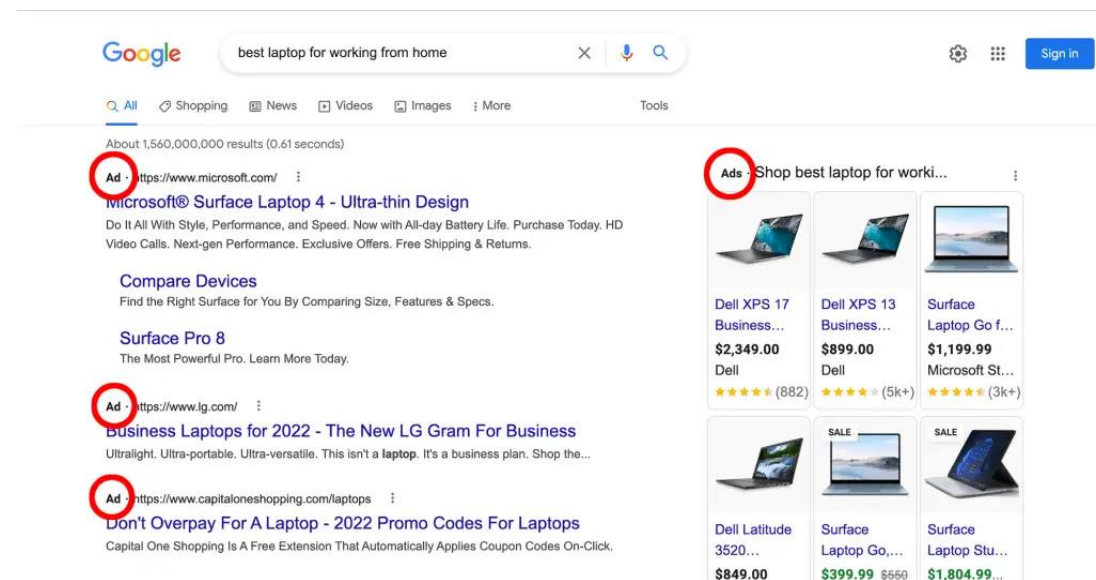
Чтобы что-то улучшить – сначала надо это что-то измерить

Друкер, Питер Фердинанд (теоретик менеджмента,
[википедия](#)) (*)

* Эту цитату еще часто приписывают экономисту Эдвардсу Демингу, физику лорду Кельвину и другим

Что такое хороший поиск?

- Очень субъективно!
- Хороший для кого: бизнеса или пользователя?
- Разным пользователям нужно разное
- Влияют география, пол, возраст, ...
- Кто-то хочет искать точно по ключевым словам, а кто-то хочет чтобы поисковик пытался «читать мысли»
- Надо как-то формализовать «хорошесть»



Google становится все хуже и хуже?

<https://nypost.com/2022/05/01/google-critics-say-ads-spam-sites-are-killing-search/>

Поведенческие метрики

- Способ №1: поведенческие метрики
- Собираем поведенческую статистику: клики, просмотры, добавления в друзья, заказы, подписки, ...
- Ставим АБ эксперименты
- Больше кликают/лайкают/покупают/... — это хорошо! (точно?)
- Дальше говорить про все это НЕ будем!

Метрика	Контроль, abs	Тест, abs	Δ, abs	Δ, %	p-value
Share_searchers_with_views30	0.13	0.12	-0.0065	-4.97%	0
Share_searches_no_click	0.63	0.65	+0.0196	+3.1%	0
Share_searches_with_click	0.37	0.35	-0.0196	-5.3%	0
Share_searches_with_click_in_top_1	0.08	0.07	-0.004	-5.21%	0
Share_searches_with_click_in_top_10	0.20	0.19	-0.013	-6.36%	0

Типичные результаты типичного АБ: новые модели просадили долю запросов с кликами

Релевантность

- Способ №2: оценка релевантности
- Релевантный документ удовлетворяет информационную потребность пользователя
- Информационная потребность может быть неоднозначна, напр. запрос «ягуар»
- Релевантность обычно оценивают с помощью *асессорской разметки*



Обязательная картинка с разными Ягуарами.

Асессорская оценка

- Используются краудсорсинговые платформы, такие как Яндекс Толока
- В компании VK есть своя in house платформа SQ (Search Quality)
- Асессоры получают задания, напр. «Оцените насколько документ D релевантен запросу Q»
- Оценки релевантности могут быть бинарными: 0 – не релевантно, 1 – релевантно
- А могут быть graded (как в школе), например:
 - 1 – нерелевантный документ, напр. *yandex.ru* по запросу ВК
 - 2 – полезный документ, напр. *VK Музыка* по запросу ВК
 - 3 – релевантный документ, напр. *vk.com* по запросу ВК

насекомые синий трактор ВСЕ ЗАПРОСЫ

ВЫБОРКА НА GOGO.ST

[youtube new] sm6, best vk model #70

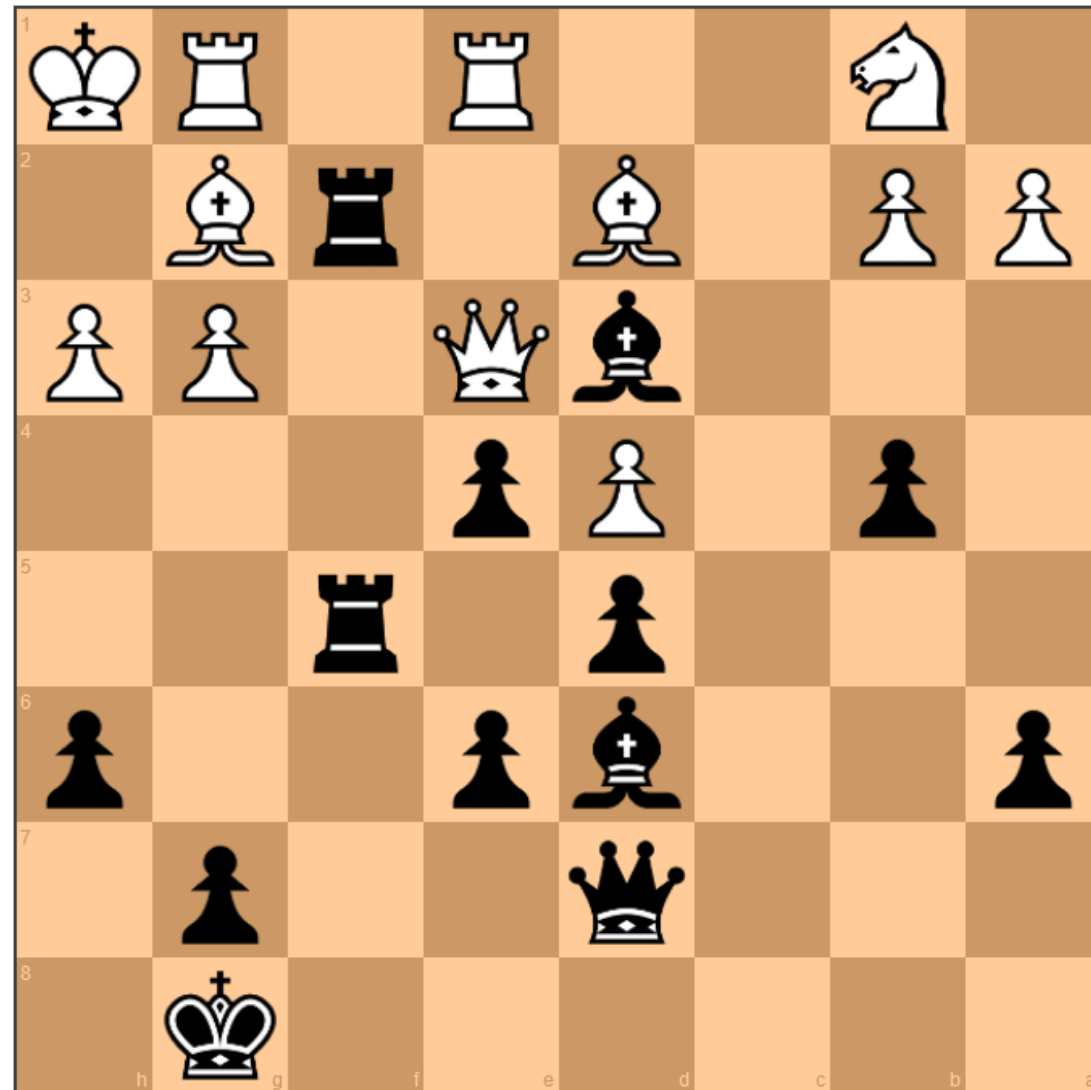
V-LDCG-10 = 19.78

Позиция	Серп	Оценка
1	https://vk.com/video148631493_456239808	2
2	https://vk.com/video-211391900_456239201	3
3	https://vk.com/video-85859225_456240469	3
4	https://vk.com/video-7966621_456239049	1
5	https://vk.com/video217045772_456239072	2
6	https://vk.com/video-221593574_456239210	2
7	https://vk.com/video1218682_456240188	3
8	https://vk.com/video536711804_456239047	2
9	https://vk.com/video59161431_456239984	2
10	https://vk.com/video176207939_456239071	3

Пример оценки релевантности по 3х-балльной шкале на платформе SQ

Проблемы оценки релевантности

- Релевантность – не панацея
- Информационная потребность может быть неоднозначна (запрос «ягуар»)
- Проблема «понимания»: ассессоры часто не разбираются в тематике поискового запроса (напр. что-то про шахматы как в примере на картинке) => не могут адекватно оценить релевантность документа запросу
- Проблема «авторитетности»: ассессоры часто не могут оценить популярность и авторитетность источника данных в области, в которой не разбираются (точно ли можно верить этому сайту про медицину?)
- Тем не менее, дальше будем работать с релевантностью



Насколько эта картинка релевантна запросу «цугцванг»?

(Из партии Земиш – Нимцович, 1923 г.)

Метрики качества

- Имея оценку, можно считать метрики качества поиска
- Их существует огромное множество: precision, recall, MAP, DCG, NDCG, MRR, ERR, pFound, RBP, ...
- Будем различать 2 случая: метрики для бинарной и метрики для graded релевантности

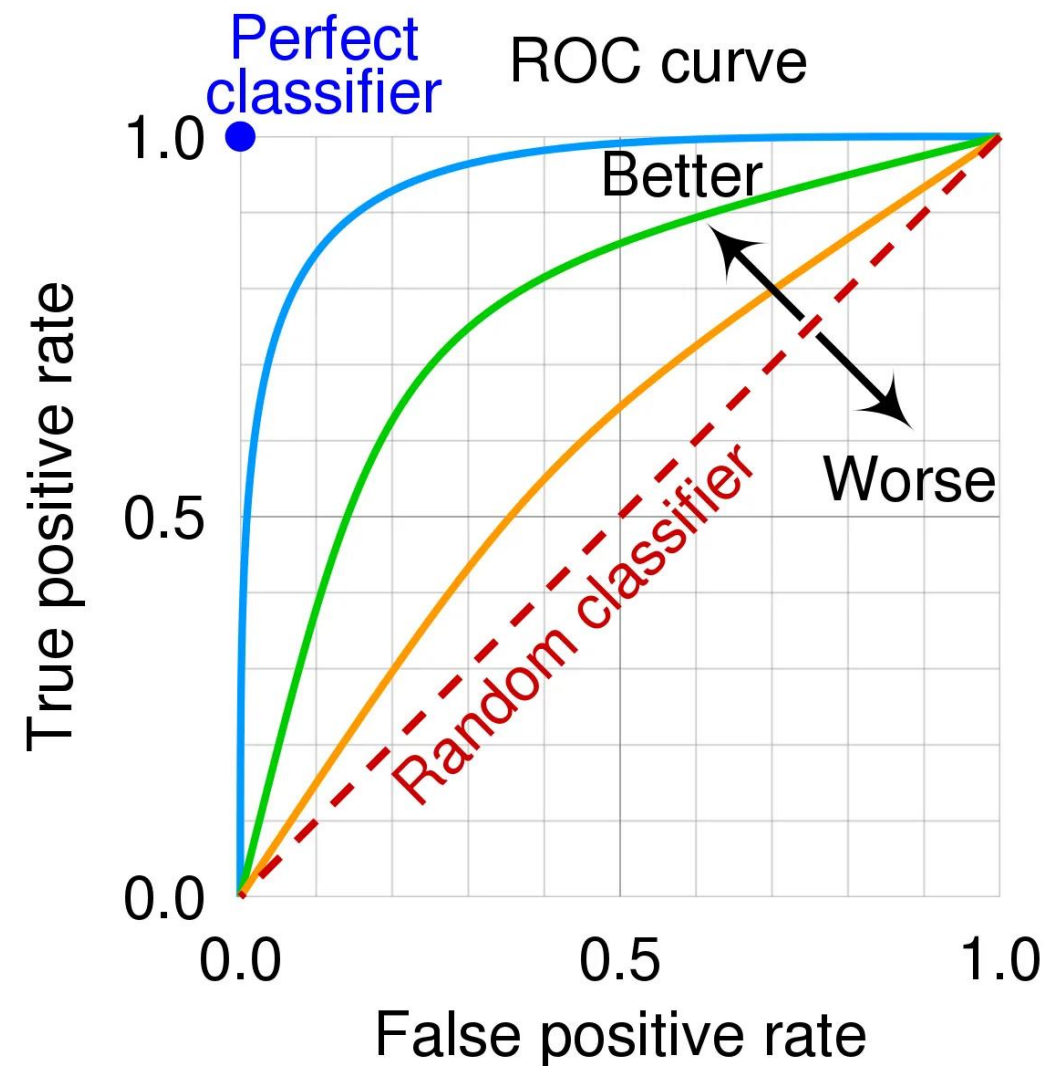
Mail sq model корзина youtube_
video exp #34

Метрика	good/total	Value
V-AdjDCG-10	626/626	6.2794
V-AdjDCG-5	626/626	4.2957
V-DCG-10	626/626	24.7909
V-DCG-5	626/626	16.7786
V-LDCG-10	626/626	24.5212
V-LDCG-5	626/626	16.6271
V-good3-SerpShare10	626/626	0.4169
V-good3-SerpShare5	626/626	0.4822
V-irrelevant-SerpShare10	626/626	0.07
V-irrelevant-SerpShare5	626/626	0.1117
V-nDCG-10	626/626	0.8529
V-nDCG-5	626/626	0.8971
V-p1	626/626	2.4357
V-good3-DocShare1		0.4871
V-good3-DocShare5		0.4474
V-good3-DocShare10		0.4169
V-good2-DocShare1		0.7283

Пример метрик, посчитанных на основе
ассессорской оценки в поиске VK

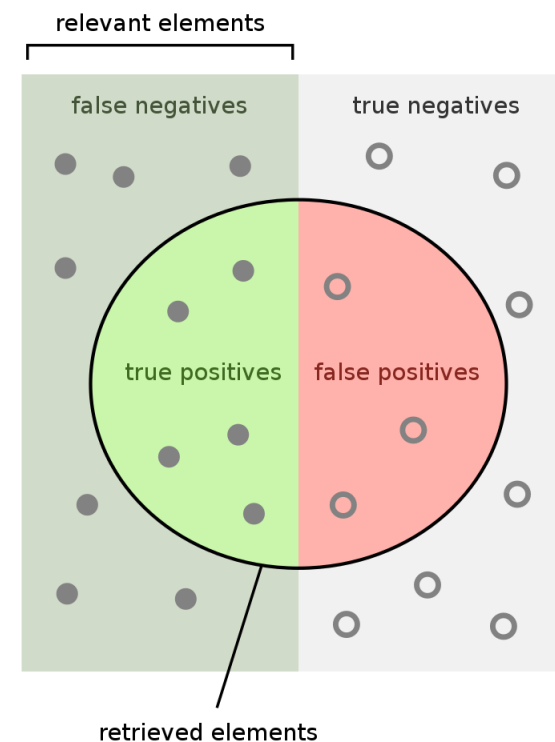
Бинарная релевантность

- Бинарная релевантность – это, по сути, классификация
- Какие вы знаете метрики качества классификации?

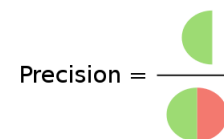


Метрики качества бинарной классификации

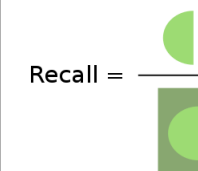
- «Аккуратность»: $Accuracy = \frac{Correct\ Classifications}{All\ Classifications}$
- Точность: $Precision = \frac{True\ Positives}{Predicted\ Positives}$
- Полнота: $Recall = \frac{True\ Positives}{All\ Positives}$
- F1-мера: $F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$
- ROC AUC и многие другие



How many retrieved items are relevant?



How many relevant items are retrieved?



Точность и полнота:

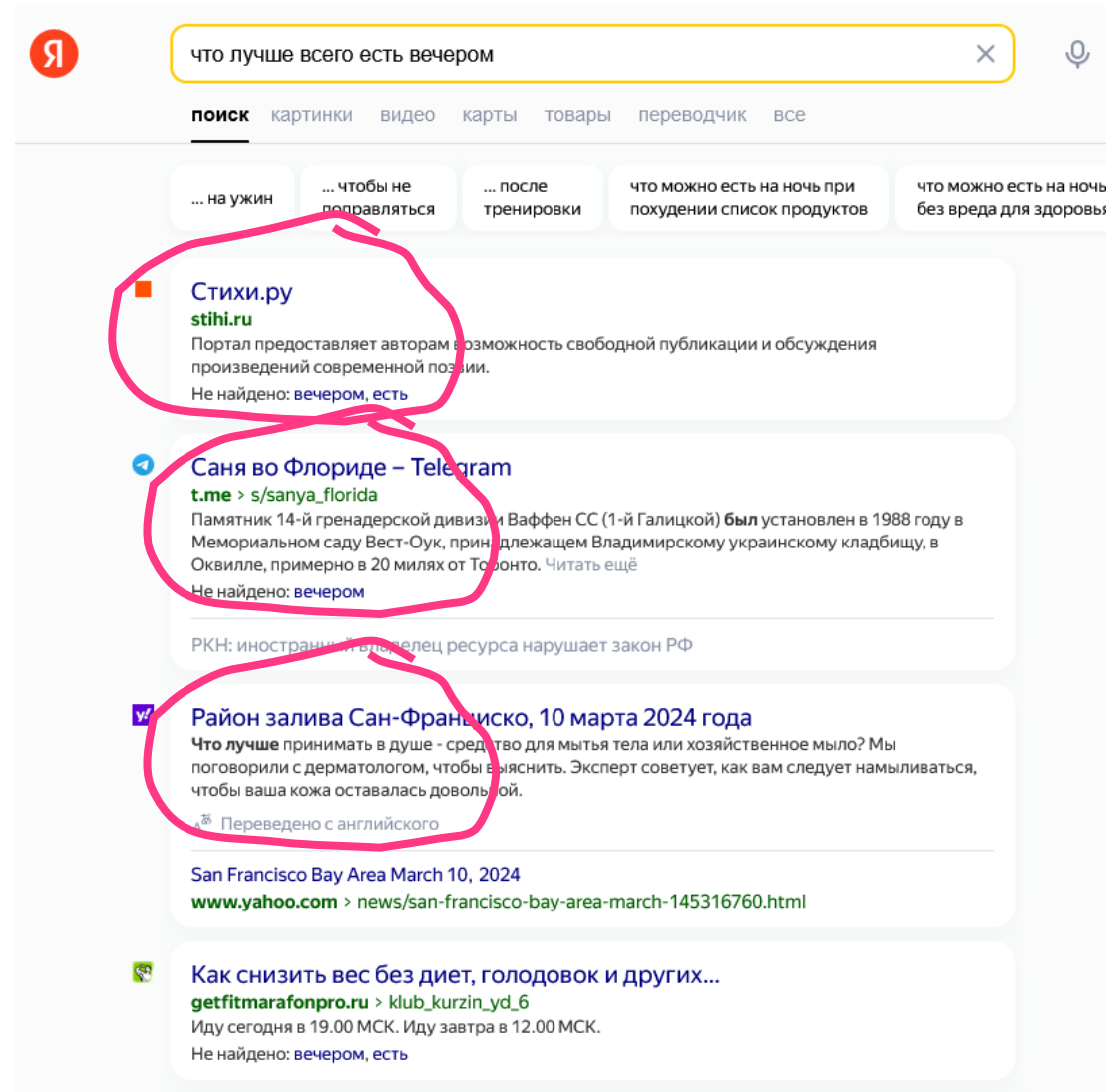
https://en.wikipedia.org/wiki/Accuracy_and_precision

Точность и полнота поиска

- В поиске огромный дисбаланс классов (число релевантных документов << числа нерелевантных) => обычно смотрят на *точность* и *полноту*
- Точность и полноту надо адаптировать под поиск
- Вместо позитивных примеров (true positives) у нас теперь релевантные документы, а вместо предсказанных позитивных примеров (predicted positives) – найденные документы
- Точность: $Precision = \frac{NumFoundRelevantDocuments}{NumFoundDocuments}$
- Полнота: $Recall = \frac{NumFoundRelevantDocuments}{NumTotalRelevantDocuments}$

«Проблема глубины»

- В «обычном» машинном обучении нам как правило известны метки классов для ВСЕГО датасета
- В поиске мы, как правило, можем оценить только топ поисковой выдачи (и не можем оценивать миллиарды документов на каждый запрос!)
- Не знаем полного числа релевантных документов => непонятно как считать полноту
- Как правило, нас интересует только качество топа (напр. Топ-10) т.к. это то что видит пользователь



Выдача 20-й страницы поиска Яндекса по запросу «что лучше всего есть вечером» - но это почти никто не видит!

(актуально на 04.2024)

Метрики@K

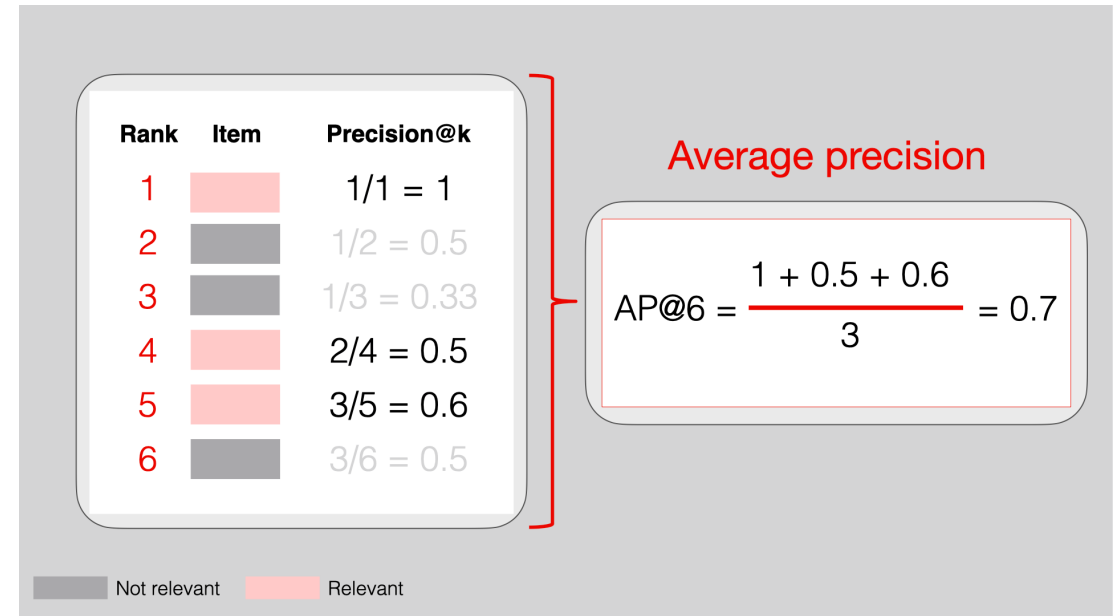
- Поисковые метрики обычно «привязывают» к глубине K, как правило K=10
- На практике обычно смотрят на $Precision@K$
- $Precision@K = \frac{NumRelevantDocuments@K}{K}$
- Ситуация осложняется тем, что поисковая система можно вернуть число документов $F < K$ (в этом случае можно поделить не на K а на F – число реально найденных документов)
- И мы пока еще не учитываем порядок документов в выдаче (см. картинку!)



У обеих моделей одинаковый $Precision@10$ но есть нюанс...
(картинка взята с evidentlyai.com)

Метрики AP@K и MAP@K

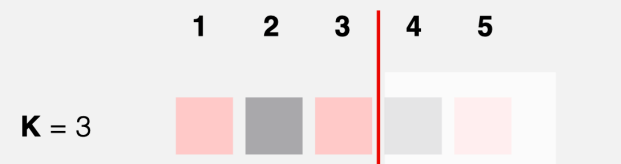
- Позиция имеет значение!
- *Average Precision* (AP@K) – расширение идеи бинарной релевантности на задачу ранжирования
- $AP@K = \frac{1}{N} \sum_{k=1}^K Precision@k * Rel(k)$ где
 - N – полное число релевантных (т.е. с лейблом 1) док-тов
 - $Rel(k)$ – релевантность документа на k-ой позиции (0 или 1)
 - Если вернули меньше K документов то «забиваем» нулями
- $MAP@K$ – это просто $AP@K$ усредненное по всем поисковым запросам
- Свойства *Average Precision*:
 - Стремится к 1 в случае полностью релевантной выдачи
 - Штрафует релевантные документы на более низких позициях



Пример расчета AP@K (картинка взята с evidentlyai.com)

Метрика DCG

- А как быть в случае graded-релевантности?
- Обычно используется метрика $DCG@K$ (Discounted Cumulative Gain)
- $DCG@K = \sum_{k=1}^K \frac{2^{Rel(k)} - 1}{\log_2(1+k)}$ где
 - $Rel(k)$ – релевантность документа на k -ой позиции, может лежать, например, в интервале $[1, 5]$ если используется 5-балльная шкала
- Свойства $DCG@K$:
 - Штрафует (discounts) вклад (gain) релевантных документов на более далеких позициях
 - Вклад равен 0 в случае если $Rel(k) == 0$ – поэтому если поиск выдал $\leq K$ документов, то просто «забываем» нулями все позиции вплоть до K -ой.



$DCG@3 = \frac{1}{\log_2(1+1)} + \frac{0}{\log_2(1+2)} + \frac{1}{\log_2(1+3)}$

$DCG@3 = 1 + 0 + 0.5 = 1.5$

Пример расчета $DCG@3$ (картинка взята с сайта evidentlyai.com).

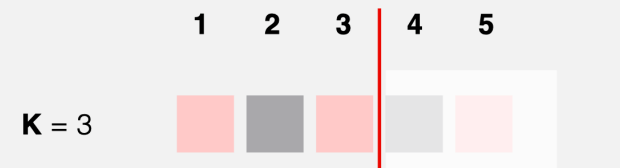
В этом примере оценки могут быть только 0 или 1 (т.е. DCG вполне применим и в случае бинарной релевантности)



Метрика NDCG

- $DCG@10 = 10$ – это хорошо или плохо?
- Проблема: значение DCG не имеет «физического» смысла и зависит от запроса, а также от шкалы оценок
- $DCG@K$ обычно нормализуют на максимально возможный $MAXDCG@K$, который получился бы, если бы мы пересортировали поисковую выдачу в порядке убывания релевантности (*)
- Определим: $NDCG@K = \frac{DCG@K}{MAXDCG@K}$
- Свойства:
 - У идеальной выдачи $NDCG@K \rightarrow 1$
 - Чем больше $NDCG@K$ – тем лучше!
 - Теперь можно сравнивать качества ранжирования на разных запросах
 - Есть подвох: если в выдаче все документы плохие (напр. все оценки 1), то $NDCG@K \rightarrow 1$ (**)

(*) (**) Тонкий момент: по-хорошему, надо пересортировывать не только выдачу, а вообще ВСЕ документы, но релевантность ВСЕХ документов нам как правило не известна


$$DCG@3 = \frac{1}{\log_2(1+1)} + \frac{0}{\log_2(1+2)} + \frac{1}{\log_2(1+3)}$$
$$DCG@3 = 1 + 0 + 0.5 = 1.5$$

Пример расчета $NDCG@3$

$$DCG@3 = 1.5$$

$$MAXDCG@3 = 1 + 1 / \log(1+2) + 0 = 1.63$$

$$NDCG@3 = 1.5 / 1.63 = 0.92$$



Метрики: recap

- Вы написали свой поисковик, как теперь оценить релевантность вашей поисковой выдачи?
- Первый делом соберите небольшой (скажем 1000) набор запросов, по которым вы хотите оценивать качество. Это могут быть, например, запросы рандомно сэмплированные из логов вашего поисковика.
- Обкачайте по этим запросам топ-K вашей поисковой выдачи и оцените каждую пару запрос-документ ассессорами. Всего таких пар у вас будет $1000 \times K$.
- Посчитайте метрики качества
- Если у вас бинарная оценка релевантности – смотрите на $MAP@K$
- Если вы используете *graded* релевантность – смотрите на $DCG@K$ и $NDCG@K$
- Оцененные пары запрос-документ вам еще пригодятся в будущем, например в качестве датасета для обучения моделей!

Вероятностные модели



Можно ли сделать лучше TF-IDF?

- До сих пор наши подходы к текстовому ранжированию были не очень строгими
- Осталось много вопросов, например:
 - Почему в формуле IDF используется именно логарифм?
 - Насколько ранжирование по TF-IDF оптимально, можно ли сделать лучше?
 - А если нет, то строго доказать что нельзя?
- Хочется подвести под все это какое-то теоретическое обоснование

Вероятностный подход

- Попробуем подойти к задаче ранжирования более формально
- Нам поможет *теория вероятностей*
- Будем говорить о вероятности релевантности R документа D по запросу Q
- Рассматриваем только случай бинарной релевантности, т.е. $R = 0$ или $R = 1$
- Попробуем моделировать вероятность $P(R=1 | Q, D)$

Почему вероятности?

- Вероятностное моделирование хорошо работает в задачах где есть естественная неопределенность
- Задача ранжирования как раз относится к таким задачам
- Релевантность субъективна: разные ассессоры могут оценить один и тот же документ по-разному
- Оцениваем неполной информацией (запрос «ягуар» – это животное или машина?)
- Все, что мы можем – это пытаться предсказывать вероятность того, что документ будет релевантен запросу, и надеяться угадать

Collection	Consistent	Inconsistent	(% of Total)
T123	689	120	15%
T6	228	45	16%
T78	308	71	19%
WT10G-binary	1413	293	17%
GOV2-binary	10138	2346	19%
WT10G-trinary	1374	332	19%
GOV2-trinary	9504	2980	24%

Table 1: Consistency of relevance judgments across TREC collections. The columns show a count of the number of pairs of duplicate documents that were judged consistently and inconsistently, followed by the percentage proportion of inconsistent pairs. Only duplicates where at least one document was judged to be relevant are included.

Один и тот же ассессор в 25% случаев оценивает один и тот же документ по-разному!

Из статьи Scholer, Falk, Andrew Turpin, and Mark Sanderson.
"Quantifying test collection quality based on the consistency of relevance judgements." 2011. ([ссылка](#))

Немного формальностей

- R , Q и D – это случайные величины
- R (с какой-то вероятностью) принимает значения 0 или 1
- D определена на множестве всех возможных документов, а Q – на множестве всех возможных запросов
- Позже мы формализуем понятие «множество всех возможных текстов»
- Можно говорить о совместном распределении $P(R, Q, D)$
- Задача сводится к тому, что нам надо как-то оценить $P(R=1 | Q, D)$

Случайная величина — переменная, значения которой представляют собой численные исходы некоторого случайного феномена или эксперимента.

Другими словами, это численное выражение результата случайного события. Если определять случайную величину более строго, то она является функцией $y = \xi(\omega)$, значения y которой численно выражают исходы ω случайного эксперимента.

(определение из [википедии](#))

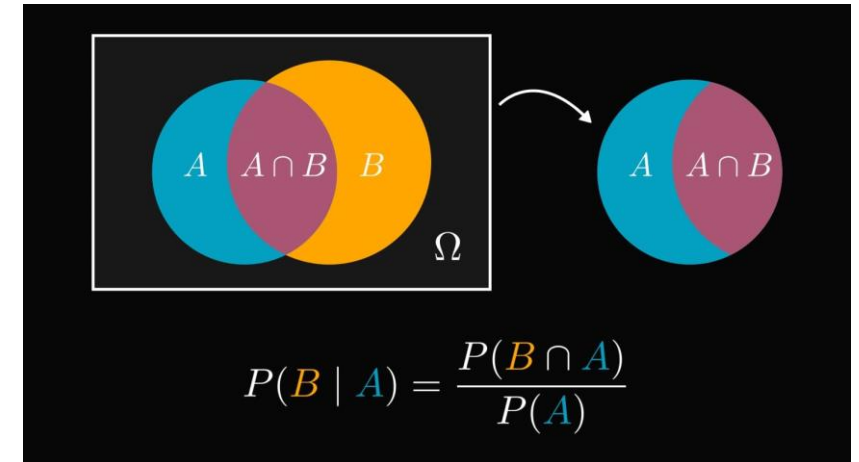
Нотация

- Мы привыкли что функции обозначаются как $f(x)$
- В теории вероятностей все не как у людей ☹
- Встречается запись $P(X)$, $p(X)$, $p(X=x)$, $p_x(x)$ и т.п.
- Такая многообразие существует не просто так, а отражает ряд важных нюансов, которыми мы, само собой, будем пренебрегать
- В нашей нотации:
 - $P(X)$ – это просто функция от случайной величины X , т.е. аналог $f(x)$
 - $P(X = x)$ – это функция, в которую подставили какое-то конкретное значение x (аналог $f(x_0)$) т.е. значение функции $f(x)$ в какой-то конкретной точке x_0)
 - $P(x)$ – это просто краткая запись $P(X = x)$
 - Другими словами, большими буквами (X) у нас переменные, а маленькими (x) – константы, т.е. конкретные значения которые принимают наши переменные



Совместная и условная вероятность

- Мы будем работать с совместными и условными распределениями нескольких случайных величин
- Совместное распределение будем записывать как $P(X, Y)$
- Условные распределения будем записывать как $P(X/Y)$
- Они связаны через т.н. product rule: $P(X, Y) = P(X/Y)P(Y)$
- Заметим, что в нашей нотации $P(X/Y)$ – это тоже функция, просто уже от двух случайных величин X и Y
- А вот $P(X = x/Y)$ – это уже функция от одной переменной Y , т.к. переменная X была зафиксирована подстановкой значения $X = x$
- Также, совместные распределения могут быть записаны в векторном виде, например $P(\mathbf{X})$ что равносильно совместному распределению нескольких скалярных случайных величин $P(X_1, X_2, X_3, \dots)$



Условные вероятности событий A и B.

Не забываем, что случайные переменные и события связаны: каждое значение случайной величины (напр. $X = 1$) определяет событие в пространстве элементарных исходов!

Принцип вероятностного ранжирования

- Допустим, мы каким-то образом оценили $P(R=1|Q,D)$
- Как теперь ранжировать документы?
- Очевидное решение: просто отсортируем все документы в порядке убывания вероятности их релевантности запросу
- Это решение называется принципом вероятностного ранжирования (*Probability Ranking Principle*, или *PRP*)
- Был предложен ван Рийсбергенем в 1979 г.
- Можно доказать что такое ранжирование оптимально (при условии что вероятности оценены настолько хорошо насколько это возможно на базе имеющихся данных)



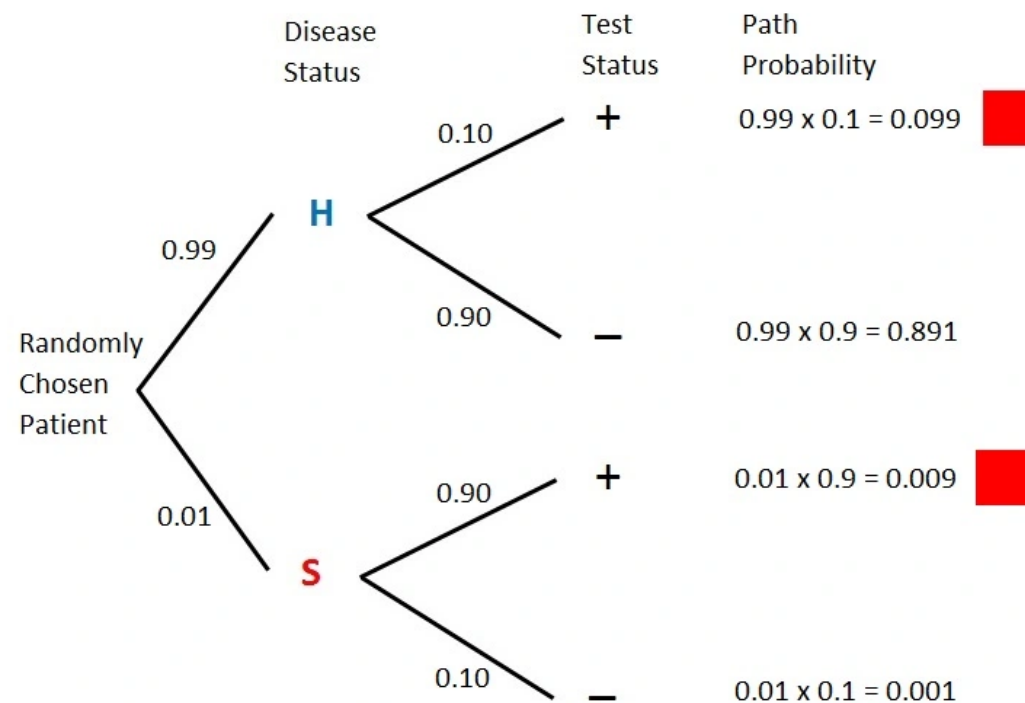
Ван Рийсберген (van Rijsbergen) (1943 г.) – один из основателей науки об информационном поиске ([википедия](#))

А еще он известен как человек который первым предложил название метрики классификации «F1-мера».

Теорема Байеса:

МОТИВАЦИЯ

- Чтобы двигаться дальше нам придется вспомнить теорему (формула) Байеса!
- Классический пример-мотивация:
 - Анализ на болезнь D показал у (случайного) пациента положительный результат
 - Какова вероятность что пациент действительно болен болезнью D?
- Заведем две бинарные случайные величины, T и D :
 - $P(T=1)$ – вероятность того, что у случайного пациента анализ дал положительный результат (это т.н. *наблюдаемая переменная*)
 - $P(D=1)$ – вероятность того, что случайный пациент болен болезнью D (это т.н. *скрытая переменная*)



Типичное дерево рассуждений в задаче медицинской диагностики

Теорема Байеса: продолжение

- Теперь мы можем моделировать зависимость между D и T
- В частности, существует какое-то (неизвестное нам) совместное распределение $P(T,D)$
- Если бы мы знали $P(T,D)$ то мы бы знали все!
- Нам надо как-то оценить $P(D=1/T=1)$ – вероятность того, что пациент действительно болен при условии, что анализ дал положительный результат
- Как это сделать?

Теорема Байеса: применение

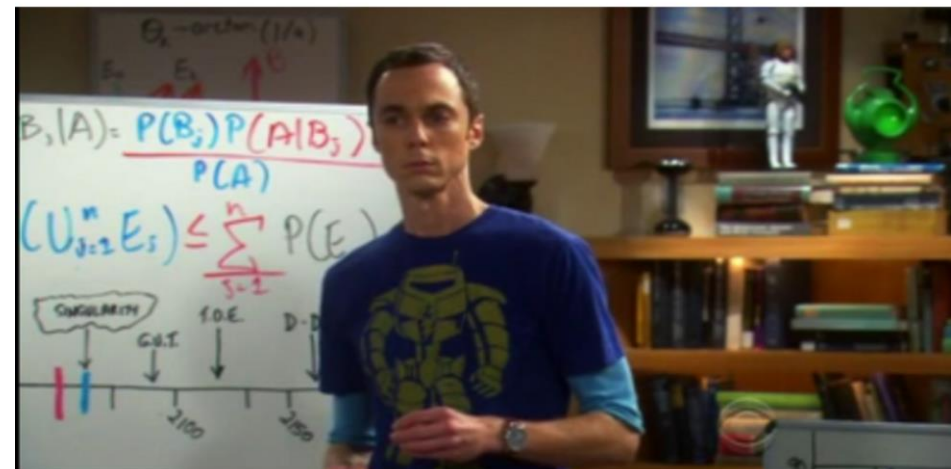
- Воспользуемся формулой Байеса для оценки $P(D/T)$: $P(D = 1|T = 1)$

$$\frac{P(T = 1|D = 1)P(D=1)}{P(T=1)} = \frac{P(T = 1|D = 1)P(D=1)}{P(T = 1|D = 1)P(D=1) + P(T = 1|D = 0)P(D=0)}$$

- Теперь нам надо откуда-то взять $P(D=1)$, $P(D=0)$, $P(T=1|D=1)$ и $P(T=1|D=0)$

- Эти величины как правило или известны, или их легко оценить по на нашему датасету (как именно – обсудим позже)

- Теорема Байеса не несет доп. информации, но позволяет «перевернуть» задачу так, чтобы неизвестная (*скрытая*) величина теперь выражается через известные, или легко поддающиеся оценке (другими словами, через *наблюдаемые* величины)



Простой вывод Т.О.Е. из формулы Байеса

Сведем задачу к предыдущей

- Вернемся к задаче оценки релевантности $P(R=1|Q,D)$
- Можно ли свести эту задачу к предыдущему примеру?
- Релевантность – это аналог неизвестной болезни
- Но что является аналогом позитивного теста, который свидетельствует о наличии болезни?
- Предположение: *сигналом о том, что документ релевантен запросу, является факт наличия (или отсутствия) в документе ключевых слов запроса*

Слова-как-сигналы:

- Если в документе есть термин МГУ – он с большой вероятностью релевантен запросу «мгу»
- Если в документе есть термин МОСКВА – тоже, возможно, релевантен
- Если есть термин ГОРОД – то сложно что-то сказать, это скорее нейтральный сигнал
- А если есть термин НОВОСИБИРСК – скорее всего, не релевантен (но не факт!)

Формализуем Q и D

- Будем рассматривать документы и запросы как «мешки» слов (терминов)
- Будем моделировать факт наличия слова с помощью бинарной случайной величины
- Пусть V – это словарь всех известных нам терминов, которые могут встречаться в документах и запросах
- Введем векторные случайные величины размером $|V|$:
 - Для запроса: $\mathbf{Q} = (Q_1, Q_2, \dots, Q_{|V|})$ где $Q_i = 1$ если i -й термин встретился в запросе
 - Для документа: $\mathbf{D} = (D_1, D_2, \dots, D_{|V|})$ где $D_i = 1$ если i -й термин встретился в документе
- Можно понимать как генеративную модель: для каждого термина из словаря подкидываем «нечестную» монетку, которая определяет есть или нет данный термин в данном документе (или запросе)
- Понятие «множество всех возможных документов/запросов» теперь обрело смысл!

Соберем все вместе

- Моделируем запросы и документы случайными векторами терминов \mathbf{Q} и \mathbf{D}
- Допустим, нам известны запрос $\mathbf{q} = (q_1, \dots, q_{|V|})$ и документ $\mathbf{d} = (d_1, \dots, d_{|V|})$
- Задача: надо оценить вероятность релевантности $P(R=1 | \mathbf{D}=\mathbf{d}, \mathbf{Q}=\mathbf{q})$
- Заметим, что тогда вероятность не-релевантности: $P(R=0 | \mathbf{D}=\mathbf{d}, \mathbf{Q}=\mathbf{q}) = 1 - P(R=1 | \mathbf{D}=\mathbf{d}, \mathbf{Q}=\mathbf{q})$
- Воспользуемся формулой Байеса чтобы «развернуть» задачу так, чтобы наша неизвестная величина легко выражалась через величины, которые легко оценить по данным

Naïve Bayes

- Спойлер: дальнейшее изложение очень похоже на наивный байесовский классификатор (модель Бернулли)!
- Только чуть сложнее, т.к. у нас есть не только метка класса и текст, а 3 сущности: метка класса (релевантность), текст запроса и текст документа
- Можно считать, что мы классифицируем на релевантность пару запрос-документ

```
TrainBernoulliNB( $\mathbb{C}$ ,  $\mathbb{D}$ )
1   $V \leftarrow \text{ExtractVocabulary}(\mathbb{D})$ 
2   $N \leftarrow \text{CountDocs}(\mathbb{D})$ 
3  for each  $c \in \mathbb{C}$ 
4  do  $N_c \leftarrow \text{CountDocsInClass}(\mathbb{D}, c)$ 
5       $\text{prior}[c] \leftarrow N_c / N$ 
6      for each  $t \in V$ 
7      do  $N_{ct} \leftarrow \text{CountDocsInClassContainingTerm}(\mathbb{D}, c, t)$ 
8           $\text{condprob}[t][c] \leftarrow (N_{ct} + 1) / (N_c + 2)$ 
9  return  $V, \text{prior}, \text{condprob}$ 

ApplyBernoulliNB( $\mathbb{C}, V, \text{prior}, \text{condprob}, d$ )
1   $V_d \leftarrow \text{ExtractTermsFromDoc}(V, d)$ 
2  for each  $c \in \mathbb{C}$ 
3  do  $\text{score}[c] \leftarrow \log \text{prior}[c]$ 
4      for each  $t \in V$ 
5      do if  $t \in V_d$ 
6          then  $\text{score}[c] += \log \text{condprob}[t][c]$ 
7          else  $\text{score}[c] += \log (1 - \text{condprob}[t][c])$ 
8  return  $\arg \max_{c \in \mathbb{C}} \text{score}[c]$ 
```

Рис. 13.3. Наивный байесовский алгоритм (модель Бернулли): обучение и тестирование. В строке 8 (вверху) применено сглаживание добавлением единицы, аналогичное формуле (13.7), где $B = 2$

Алгоритм наивного байесовского классификатора (из книги Маннинга)

Датасет

- После применения формулы Байеса нам придется оценивать наблюдаемые переменные по данным
- Будем считать, что у нас есть датасет, который состоит из:
 - Корпуса текстовых документов
 - Набора запросов
 - И нам известна релевантность (0 или 1) для всех пар запрос-документ (ВАЖНО: на практике такое бывает редко и мы еще вернемся к этому моменту)
- А $P(R=1 | \mathbf{D}, \mathbf{Q})$ – это по сути модель (очень похожая на наивный байесовский классификатор), которую мы хотим «обучить» на этом датасете

Напоминает постановку типичной задачи машинного обучения, не так ли?

Пример датасета

- Корпус из 3х документов:
 1. *Московский физико-технический институт*
 2. *Московский государственный университет*
 3. *Университет ИТМО*
- Даны 2 запроса:
 1. *МФТИ*
 2. *университет*
- Даны оценки релевантности пар запрос-документ:
 - $Q1:D1 \rightarrow 1, Q1:D2 \rightarrow 0, Q1:D3 \rightarrow 0$
 - $Q2:D1 \rightarrow 0, Q2:D2 \rightarrow 1, Q2:D3 \rightarrow 1$

Используем бинарные оценки релевантности!

Решающее правило

- На практике удобнее работать не с вероятностями классов $P(R=1 | \mathbf{D}, \mathbf{Q})$ а с их отношениями
- Определим решающее правило $O(\mathbf{D}, \mathbf{Q}) = \frac{P(R = 1 | \mathbf{D}, \mathbf{Q})}{P(R = 0 | \mathbf{D}, \mathbf{Q})}$
- Если мы отсортируем документы по $O(\mathbf{d}, \mathbf{q})$ то получим такой же порядок как если бы мы отсортировали по $P(R=1 | \mathbf{d}, \mathbf{q})$
- Задача свелась к оценке $O(\mathbf{d}, \mathbf{q})$
- Логарифм такого отношения вероятностей часто называют *логитами*

Теперь пойдут выкладки...

- Применим формулу Байеса и получим вероятности релевантности:

- $$P(R = 1|\mathbf{D}, \mathbf{Q}) = \frac{P(\mathbf{D}|R = 1, \mathbf{Q})P(R = 1|\mathbf{Q})}{P(\mathbf{D}|\mathbf{Q})}$$

- $$P(R = 0|\mathbf{D}, \mathbf{Q}) = \frac{P(\mathbf{D}|R = 0, \mathbf{Q})P(R = 0|\mathbf{Q})}{P(\mathbf{D}|\mathbf{Q})}$$

где сумма $P(R = 1|\mathbf{D}, \mathbf{Q}) + P(R = 0|\mathbf{D}, \mathbf{Q}) = 1$

- Подставим в решающее правило, и заметим что мы всегда ранжируем документы для фиксированного запроса:

$$O(\mathbf{D}, \mathbf{Q} = \mathbf{q}) = \frac{P(R = 1|\mathbf{D}, \mathbf{Q} = \mathbf{q})}{P(R = 0|\mathbf{D}, \mathbf{Q} = \mathbf{q})} = \frac{P(R = 1|\mathbf{Q} = \mathbf{q})}{P(R = 0|\mathbf{Q} = \mathbf{q})} * \frac{P(\mathbf{D}|R = 1, \mathbf{Q} = \mathbf{q})}{P(\mathbf{D}|R = 0, \mathbf{Q} = \mathbf{q})}$$

Константа для данного запроса \mathbf{q} : $O(\mathbf{Q} = \mathbf{q})!$

... и получаем

- Приходим к формуле:

$$O(\mathbf{D}, \mathbf{Q} = \mathbf{q}) = O(\mathbf{Q} = \mathbf{q}) * \frac{P(\mathbf{D} | R = 1, \mathbf{Q} = \mathbf{q})}{P(\mathbf{D} | R = 0, \mathbf{Q} = \mathbf{q})}$$

- Чтобы двигаться дальше давайте зафиксируем какой-то конкретный документ $\mathbf{D} = \mathbf{d}$ и попробуем оценить выражение $P(\mathbf{D} = \mathbf{d} | R = 1, \mathbf{Q} = \mathbf{q})$ – вероятность того, что мы наблюдаем данный документ \mathbf{d} при условии, что нам известен запрос \mathbf{q} , и известно, что данный документ релевантен данному запросу.

Бинарная модель независимости

- Как оценить $P(\mathbf{D} = \mathbf{d} | R = 1, \mathbf{Q} = \mathbf{q})$?
- Сделаем сильное предположение: факт наличия (или отсутствия) термина в документе не зависит от наличия (отсутствия) любого другого термина в документе (при условии что запрос задан)
- Это, очевидно, в общем случае неверно (если в документе есть термин МГУ, то скорее всего есть и УНИВЕРСИТЕТ), но сильно упрощает задачу
- Такая модель называется бинарной моделью независимости (*Binary Independence Model*, или просто *BIM*)
- *BIM* – это, по сути, аналог «наивности» в наивном байесовском классификаторе!
- Наша вероятность превращается в произведение по всем терминам из словаря:

$$P(\mathbf{D} = \mathbf{d} | R = 1, \mathbf{Q} = \mathbf{q}) = \prod_{i=1}^{|\mathbf{V}|} P(D_i = d_i | R = 1, \mathbf{Q} = \mathbf{q})$$

Еще немного формул...

- С помощью предположения о бинарной независимости получим:

$$O(\mathbf{D} = \mathbf{d}, \mathbf{Q} = \mathbf{q}) = \frac{P(\mathbf{D} = \mathbf{d} | R = 1, \mathbf{Q} = \mathbf{q})}{P(\mathbf{D} = \mathbf{d} | R = 0, \mathbf{Q} = \mathbf{q})} = O(\mathbf{Q} = \mathbf{q}) * \prod_{i=1}^{|V|} \frac{P(D_i = d_i | R = 1, \mathbf{Q} = \mathbf{q})}{P(D_i = d_i | R = 0, \mathbf{Q} = \mathbf{q})}$$

- Каждое d_i это или 0 (данного термина нет в документе) или 1, поэтому мы можем их разделить:

$$O(\mathbf{D} = \mathbf{d}, \mathbf{Q} = \mathbf{q}) = O(\mathbf{Q} = \mathbf{q}) * \prod_{i:d_i=1} \frac{P(D_i = 1 | R = 1, \mathbf{Q} = \mathbf{q})}{P(D_i = 1 | R = 0, \mathbf{Q} = \mathbf{q})} * \prod_{i:d_i=0} \frac{P(D_i = 0 | R = 1, \mathbf{Q} = \mathbf{q})}{P(D_i = 0 | R = 0, \mathbf{Q} = \mathbf{q})}$$

Упрощаем...

- Для удобства введем переменные:
- $p_i = P(D_i = 1 | R = 1, \mathbf{Q} = \mathbf{q})$ – вероятность того, что i -й термин встретится в релевантном документе (при заданном запросе)
- $u_i = P(D_i = 1 | R = 0, \mathbf{Q} = \mathbf{q})$ – вероятность того, что i -й термин встретится в нерелевантном документе (при заданном запросе)
- Тогда наше выражение приобретает вид:

$$O(\mathbf{D} = \mathbf{d}, \mathbf{Q} = \mathbf{q}) = O(\mathbf{Q} = \mathbf{q}) * \prod_{i:d_i=1} \frac{p_i}{u_i} * \prod_{i:d_i=0} \frac{(1-p_i)}{(1-u_i)}$$

Еще упрощаем...

- Сделаем еще одно сильное предположение: термины, которые НЕ встречаются в запросе, с одинаковой вероятностью встречаются в релевантных и нерелевантных документах
- Другими словами, считаем что $p_i = u_i$ если $q_i = 0$
- Тогда выражение можно преобразовать к виду:

$$O(\mathbf{D} = \mathbf{d}, \mathbf{Q} = \mathbf{q}) = O(\mathbf{Q} = \mathbf{q}) * \prod_{i: d_i=1, q_i=1} \frac{p_i}{u_i} * \prod_{i: d_i=0, q_i=1} \frac{1-p_i}{1-u_i}$$

- Первое произведение вычисляется по тем терминам запроса, которые найдены в документе
- А второе по тем, которые не найдены
- ВАЖНО: мы перешли от произведения по всем терминам словаря к произведению по терминам запроса!

Преобразуем...

- Включим во второе произведение все термины запроса, которые были найдены в документе (и, одновременно, учтем их в первом произведении так, чтобы значение выражения не поменялось):

$$O(\mathbf{D} = \mathbf{d}, \mathbf{Q} = \mathbf{q}) = O(\mathbf{Q} = \mathbf{q}) * \prod_{i: d_i=1, q_i=1} \frac{p_i(1-u_i)}{u_i(1-p_i)} * \prod_{i: q_i=1} \frac{1-p_i}{1-u_i}$$

Опять константа для данного запроса \mathbf{q} !

- Первое произведение все еще только по тем терминам запроса, которые были найдены в документе
- А вот второе уже вообще по всем словам запроса – а это константа для заданного запроса!
- По сути, все что нам осталось оценить – это произведение: $\prod_{i: d_i=1, q_i=1} \frac{p_i(1-u_i)}{u_i(1-p_i)}$

RSV (Retrieval Status Value)

- Заметим, что с тем же успехом можем ранжировать по логарифму от $O(\mathbf{D} = \mathbf{d}, \mathbf{Q} = \mathbf{q})$
- Возьмем логарифм и отбросим константы $O(\mathbf{Q} = \mathbf{q})$ and $\prod_{i:q_i=1} \frac{1-p_i}{1-u_i}$
- Определим RSV (или, по-другому, *формула Робертсона / Спарк Джонс*):

$$RSV = \log \prod_{i:d_i=1, q_i=1} \frac{p_i(1-u_i)}{u_i(1-p_i)} = \sum_{i:d_i=1, q_i=1} \log \frac{p_i(1-u_i)}{u_i(1-p_i)}$$

- Или: $RSV = \sum_{i:d_i=1, q_i=1} c_i$ где мы ввели обозначение $c_i = \log \frac{p_i(1-u_i)}{u_i(1-p_i)}$
- Будем ранжировать наши документы по RSV !

Как оценить RSV?

- $RSV = \sum_{i:d_i=1, q_i=1} c_i$ где:
- $c_i = \log \frac{p_i(1-u_i)}{u_i(1-p_i)}$
- p_i – вероятность того, что i -й термин встретится в релевантном документе
- u_i – вероятность того, что i -й термин встретится в нерелевантном документе

documents	relevant	nonrelevant	total
term present: $d_i = 1$	s	$df_i - s$	df_i
term present: $d_i = 0$	$S - s$	$(N - df_i) - (S - s)$	$N - df_i$
total	S	$N - S$	N

Теперь мы можем просто оценить по датасету u_i и p_i , а значит и c_i , путем простого подсчета:

- $p_i = s/S$
- $u_i = (df_i - s)/(N - S)$
- ... и подставляем в c_i (можно еще добавить сглаживание)

Ранжируем с помощью RSV

- На входе: датасет из пар запрос-документ, и их релевантностей
- На этапе обучения:
 - Считаем число вхождений терминов запроса в релевантных и нерелевантных документах
 - Вычисляем вероятности p_i и u_i
- На этапе обработки запроса:
 - Подставляем p_i и u_i в формулу $RsvScore(q, d) = RSV = \sum_{i: d_i=1, q_i=1} \log \frac{p_i(1-u_i)}{u_i(1-p_i)}$
 - Ранжируем документы по значению $RsvScore(q, d)$

Еще немного упрощаем...

- Теперь мы можем объяснить откуда в формуле IDF взялся логарифм!
- Перепишем RSV в виде:
 - $RSV = \sum_{i:d_i=1, q_i=1} \log\left(\frac{p_i}{1-p_i} * \frac{1-u_i}{u_i}\right)$
 - где:
 - $\frac{p_i}{1-p_i}$ – это статистика посчитанная только по релевантным документам
 - $\frac{1-u_i}{u_i}$ – это статистика посчитанная только по нерелевантным документам

Круг замкнулся

- Сделаем еще одно упрощающее предположение: аппроксимируем статистику по нерелевантным документам статистикой по всей коллекции (так можно, потому что подавляющее большинство документов в коллекции являются нерелевантными)
- Пусть df_i – это число документов в которых содержится i -й термин, а N – это полное число документов
- Тогда вероятность присутствия термина в нерелевантном документе $u_i = \frac{df_i}{N}$ (и теперь она одинакова для всех запросов!)
- Тогда $\log(\frac{1-u_i}{u_i}) = \log \frac{N-df_i}{df_i} \approx \log \frac{N}{df_i}$ – вот он наш IDF!
- А RSV приобретает вид: $RSV = \sum_{i:d_i=1, q_i=1} (\log(\frac{p_i}{1-p_i}) + IDF)$
- Если все p_i одинаковы (т.е. все термины запроса имеют равные шансы появиться в релевантных документах) то наш RSV превращается в уже знакомую сумму IDFов

Недостатки RSV

- Посмотрим еще раз на финальную формулу: $RSV = \sum_{i:d_i=1, q_i=1} (\log(\frac{p_i}{1-p_i}) + IDF)$
- Проблема №1: чтобы оценить p_i все еще нужен датасет с оценками для всех пар запрос-документ!
- На практике оценить все пары невозможно, поэтому обычно оценивают сэмплы и аппроксимируют p_i только по известным релевантным документам
- Проблема №2: мы не учитываем TF => будем плохо ранжировать однословные запросы
- Можно ли пойти дальше и получить простую универсальную формулу а-ля TF-IDF в которой вообще не будет неизвестных параметров?
- Оказывается, можно

BM25

- На практике вместо *RSV* обычно используют модель *BM25*
- Ее тоже можно вывести из предположения о бинарной независимости (но мы этого делать не будем!)
- Относится к целому семейству моделей *BM* (есть еще например *BM11* и *BM15*)
- *BM* расшифровывается как Best Match
- Впервые была использована в 80-е годы в поисковом движке Окарі, поэтому известна так же как Окарі BM25



Так выглядит окапи ([википедия](#))

Ранжируем с помощью BM25

- Пусть дан запрос $q = (t_1, t_2, \dots, t_Q)$
- Тогда для каждого документа d вычислим:

$$Bm25Score(q, d) = \sum_{t_i \in q} IDF(t_i) * \frac{TF(t_i, d)(k_1 + 1)}{TF(t_i, d) + k_1(1 - b + b \frac{L(d)}{L_{avg}})}$$

- Тут:
 - $TF(t_i)$ – это частота термина t_i в документе d
 - $IDF(t_i) = \log \frac{N - DF(t_i) + 0.5}{DF(t_i) + 0.5}$ – это просто IDF (со сглаживанием) термина t_i
 - N – это полное число документов, $DF(t_i)$ – число документов которые содержат термин t_i
 - $L(d)$ – это длина d (в терминах), а L_{avg} – это средняя длина документа
 - k_1 и b свободные параметры, обычно $k_1 = 1.2$ и $b = 0.75$ (при желании их можно тюнить на валидации)
- И ранжируем документы в порядке убывания $Bm25Score(q, d)$

Свойства BM25

- Работает не только для бинарной релевантности т.к. использует TF
- Устойчиво к спаму: с ростом TF выходит на затухание
- Учитывает длину документа: штрафует слишком длинные документы
- В отличие от *RSV*, не требует подсчета зависящих от запроса статистик типа p_i (*)
- Все что нужно – это корпус текстов, по которому рассчитываются IDFы
- Очень хорошо работает на практике!
- *BM25* – это «золотой стандарт» с которым сравнивают другие алгоритмы
- Только с появлением BERTов (в 2019 г.) удалось убедительно победить *BM25*

$$Bm25Score(q,d) = \sum_{t_i \in q} IDF(t_i) * \frac{TF(t_i,d)(k_1+1)}{TF(t_i,d)+k_1(1-b+b\frac{L(d)}{L_{avg}})}$$

(*) но если вам повезло и есть полные оценки для всех пар запрос-документ, то лучше использовать *RSV*

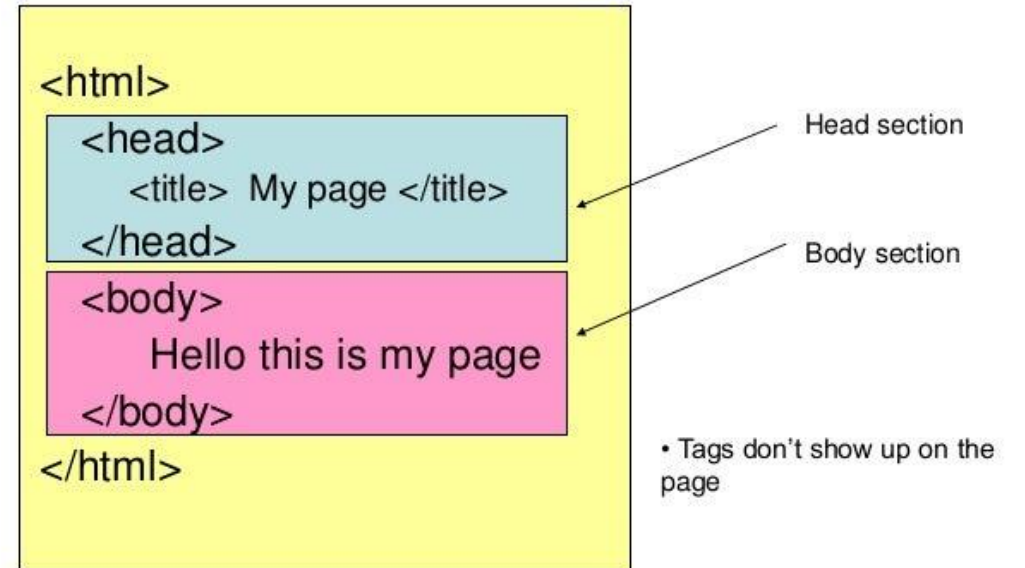
Расширения текстового ранжирования



Текстовые зоны

- Обычно текстовые документы состоят из нескольких зон
- Например, документы в нашем ДЗ будут состоять из зоны TITLE и зоны BODY
- Интуитивно кажется, что ключевое слово, найденное в зоне TITLE, является более важным «сигналом» о релевантности документа чем оно же но в зоне BODY
- Как нам модифицировать наши ранки чтобы это учесть?

Basic Web Page



В любом HTML-документе можно выделить как минимум 2 текстовые зоны: TITLE и BODY

Зонные индексы

- Сначала потребуется немного модифицировать структуру индекса
- Будем хранить информацию о зонах прямо в словопозициях
- Тогда для каждого поднятого из обратного индекса документа-кандидата нам сразу доступна информация о том, в каких зонах был найден термин
- Теперь мы можем учитывать зоны на этапе расчета текстовых рангов

130 Глава 6. Ранжирование, взвешивание терминов и модель векторного пространства

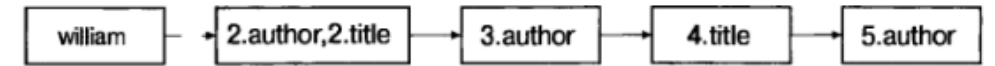


Рис. 6.3. Зонный индекс, в котором зона закодирована в словопозиции, а не в словаре

Каждая словопозиция содержит информацию о зонах, в которых в данном документе содержится данный термин.

В этом примере у нас 2 зоны: TITLE и AUTHOR, т.к. мы индексируем корпус книг (взято из Маннинга)

Зонные ранки

- Но как именно учитывать информацию о зонах?
- Напрашиваются 2 подхода
- Можно просто сконкатенировать все тексты и посчитать глобальный *BM25*
- А можно посчитать *BM25* по каждой зоне в отдельности и смешать
- Но как именно смешать?

Линейная комбинация зонных рангов

- Считаем $Bm25Score(q, z)$ для каждой зоны z документа d по отдельности
- И просто суммируем зонные BM25 с некими коэффициентами, например:

$$ZonedBm25Score(q, d) = g * Bm25Score(q, title) + (1 - g) * Bm25Score(q, body)$$

- Значения коэффициентов можно выучить на обучающем множестве

BM25F

- Оказывается, можно сделать немного лучше, если суммировать с весами не зонные ранки, а TF терминов в зонах
- Вместо обычного $TF(t, d)$ будем использовать взвешенный $TW(t, d)$
- Определим (суммируем по всем зонам z документа d): $TW(t, d) = \sum_z w_z * \frac{TF(t, z)}{1 - b_z + b_z \frac{L(z, d)}{L_{avg}(z)}}$
- $TF(t, z)$ – это обычная частота термина t в зоне z документа d
- $L(z, d)$ – это длина зоны z в документе d , а $L_{avg}(z)$ – средняя длина зоны z в коллекции
- w_z и b_z – это зонные параметры которые подбираются на обучающем множестве
- И дальше будем ранжировать документы по формуле BM25F:

$$Bm25FScore(q, d) = \sum_{t \text{ in } q} IDF(t) * \frac{TW(t, d)(k_1 + 1)}{TW(t, d) + k_1(1 - b + b \frac{L(d)}{L_{avg}})}$$

Близость и порядок слов

- Хотим как-то учитывать в ранжировании близость и порядок слов
- Мы рассмотрим простейшее решение основанное на т.н. *пассажах*
- Но сначала нам понадобится добавить в индекс информацию о позиции слов в документах

- Эти документы всегда будут иметь один и тот же $TfIdfScore(q,d)$:
 1. МФТИ лучше чем МГУ
 2. МГУ лучше чем МФТИ

Но, очевидно, смысл этих документов отличается!

Координатные индексы

- Все совершенно аналогично ситуации с зонами!
- Будем хранить информацию о позициях (координатах) терминов в документе прямо в словопозициях
- Тогда для каждого поднятого из обратного индекса документа-кандидата нам сразу доступна информация о том, на каких позициях он был найден
- Теперь мы можем учитывать позиции на этапе расчета текстовых рангов

to, 993427:

⟨1, 6: ⟨7, 18, 33, 72, 86, 231⟩;
2, 5: ⟨1, 17, 74, 222, 255⟩;
4, 5: ⟨8, 16, 190, 429, 433⟩;
5, 2: ⟨363, 367⟩;
7, 3: ⟨13, 23, 191, ...⟩; ... ⟩

be, 178239:

⟨1, 2: ⟨17, 25⟩;
4, 5: ⟨17, 191, 291, 430, 434⟩;
5, 3: ⟨14, 19, 101⟩; ... ⟩

Рис. 2.11. Пример координатного индекса. Слово to встречается в документах 993 477 раз, причем в документе 1 оно встречается на позициях 7, 18, 33 и т.д

Тут каждая словопозиция содержит:

- Как обычно, docID и TF термина в документе
- Список позиций (координат) на которых термин встречался в документе

(картинка из книги Маннинга)

Пассажи

- *Пассаж* – это просто какая-то последовательность терминов документе
- Возможно 2 подхода к выделению пассажей:
 - Просто разбиваем документ на части длиной L
 - Пробегаем по документу *скользящим окном* длины L

Документ «московский государственный университет» содержит 2 пассажа длины $L=2$ при выделении пассажей методом скользящего окна:

- московский государственный
- государственный университет

Пассажные ранки

- Как теперь использовать пассажи в ранжировании?
- Считаем ранки (напр. $Bm25Score(q,p)$) независимо для каждого пассажа p
- Опционально, добавляем бонус за правильный порядок слов в пассаже
- Как-то агрегируем ранки всех пассажей – в простейшем случае просто берем в качестве ранка всего документа лучший $Bm25Score(q,p)$ лучшего пассажа
- Таким образом мы получим высокие ранки для документов, в которых ключевые слова были расположены рядом (и в правильном порядке, если мы использовали бонус за порядок слов)

DocRank

- Объединим вместе (зонный) *BM25* и пассажный ранк:

$$DocRankScore(q, d) = g * Bm25FScore(q, d) + (1 - g) * BestPassageBm25Score(q, d)$$

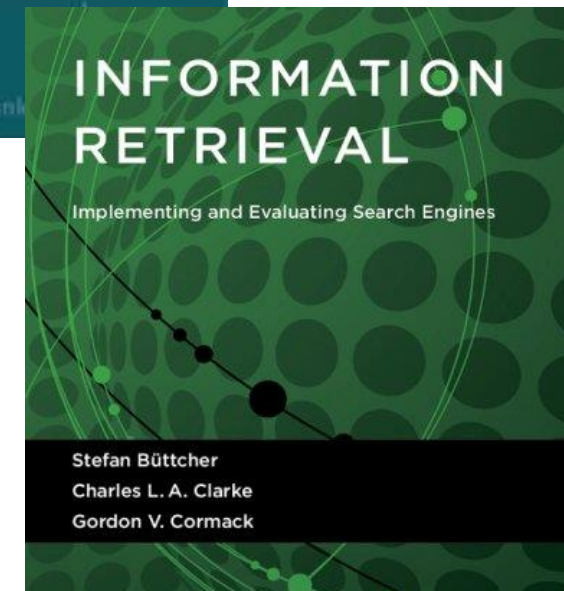
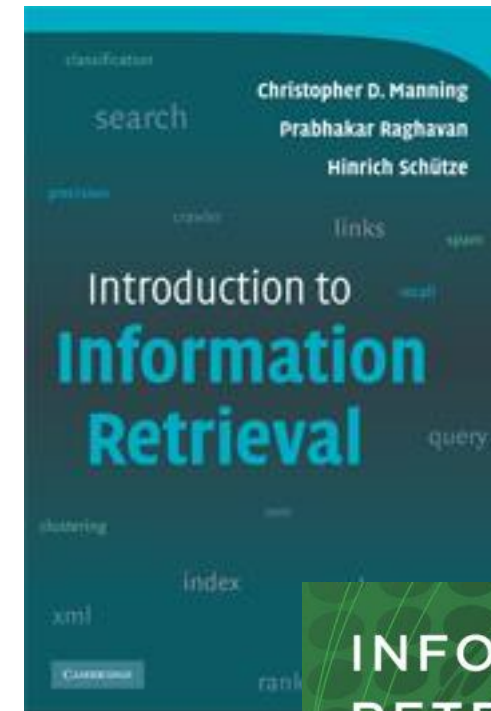
- Очень хорошо работает на практике
- Активно использовалось в Поиске@Mail.ru до 2012 года!
- Опять линейная модель – нам никуда не убежать от ML

Материалы



Книги и статьи

- Книга Маннинга: Manning, Christopher D. *An introduction to information retrieval*. 2009. ([home](#))
 - Глава 8 про метрики качества поиска
 - Глава 11 про вероятностные модели
- Книга Бюттшера: Buttcher, Stefan, Charles LA Clarke, and Gordon V. Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016 ([home](#))
 - Глава 8 про вероятностные модели
 - Глава 12 про метрики качества



Семинар и ДЗ



ДЗ

- Основано на датасете VK MARCO (разберем на семинаре)
- Оформлено как констест на Kaggle: <https://www.kaggle.com/competitions/text-ranking-homework-vk-ir-fall-2024> (первый доступ по invite link: <https://www.kaggle.com/t/51ceb80521aa4d2c99f10133dfec1931>)
- Надо ранжировать документы используя только их тексты
- Baseline: обычный $TfIdfScore(q, d)$ поверх конкатенации title + body
- Можно:
 - использовать все знания, полученные на занятиях про текстовое ранжирование, в т.ч. TF-IDF, BM25, BM25F, ...
 - учитывать порядок и близость слов и т.п.
 - использовать стемминг, лемматизацию и т.д.
- Нельзя: ML (за рамками подбора коэфф-тов)
- Тем более нельзя: нейронки

Спасибо за
внимание!

