

Информационный поиск

Константин Мосиенко

Yandex School of Data Analysis *konstantin.mosienko@gmail.com*

2017

Содержание

- 1 Введение
- 2 Модели поиска
 - Простые модели
 - Обобщенная векторная модель
 - Latent Semantic Indexing
- 3 Языковые модели, лингвистика и NLP
 - Языковые модели. Простые закономерности
 - N-граммная языковая модель
 - Сглаживание
- 4 Поисковый робот
 - Нормализация URL
 - Page Rank
 - HITS
 - Power Method
- 5 Обнаружение дубликатов
 - Контентные дубли

Введение

- C.D. Manning, P. Raghavan, H. Schutze. Introduction to Information Retrieval [2008]
- B. Croft, D. Metzler, T. Strohman. Search Engines: Information Retrieval in Practice [2009]
- S. Buttcher, C. L. A. Clarke and G. V. Cormack. Information Retrieval: Implementing and Evaluating Search Engines [2010]
- https://en.wikipedia.org/wiki/Information_retrieval

- TREC (Text Retrieval Conference)
- CLEF (Cross Language Evaluation Forum)
- WWW (World Wide Web Conference)
- ESSIR (European Summer School in Information Retrieval)
- SIGIR (Special Interest Group on Information Retrieval)
- WSDM (Web Search and Data Mining)
- CIKM (Conference on Information and Knowledge Management)

Определение

Определение

Информационный поиск – это область научных исследований, ориентированная на изучение структуры, организации, хранения, поиска и извлечения информации. [G. Salton, 1968]

Определение

Информационный поиск — процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности, и наука об этом поиске. [C. Manning, 2011]

Мы будем рассматривать вопросы, касающиеся поиска по интернет сайтам.

Запрос - документ

Определение

Документ - это информационная сущность, которая хранится в базе поисковой системы (индексе). Процесс занесения документа в индекс - индексация. Документом могут быть: локальные файлы различных форматов, html-страницы, видео, аудио, картинки.

Определение

Запрос - способ выражения информационных потребностей. Обычно запрос задаётся с помощью языка запросов соответствующей поисковой системы.

Что умеет поисковая система

- Находить и скачивать документы.
- Детектировать язык и кодировку. Извлекать информацию из документов различных форматов.
- Оценивать частоту обновления документа.
- Находить в своей базе похожие документы и спам.
- Быстро отвечать на запросы к своему индексу.
- Ранжировать результаты поиска по релевантности.

Определение

Релевантность - семантическое соответствие поискового запроса и найденного документа.

Слово «поиск» может употребляться в контексте разных задач:

- Поиск в имеющейся базе. Например, поиск релевантных запросу документов в индексе поисковой системы. Базовая операция - перечисление документов, содержащих определённое слово(словосочетание).
- Обнаружение кандидатов на занесение в индекс. Например, поиск в интернете отсутствующих в индексе(новых) документов. Базовая операция - перечисление документов, на которые есть ссылки с имеющегося документа.

Некоторые особенности и сложности

- Информация доступна в неструктурированном с точки зрения индексирования виде: например, как понять, где на странице важный текст, а где рекламный блок?
- Пользователь не всегда ищет текст, он может искать и видео.
- Актуальность. Необходимо иметь как можно более точный «слепок» интернета. Быстро находить новую информацию и не забывать удалять не актуальную.
- Региональность. Один запрос, заданный из разных мест, иногда должен приводить к разным результатам. Например, если вы заказываете пиццу.

- Поисковый робот. Скачивает документы из интернета, обнаруживает новые документы, планирует очередь скачки (так как обычно нет возможности скачать все известные документы, необходимо сделать выбор, какие обойти сейчас, а какие, может быть, никогда).
- Индексатор. Обрабатывает скаченные документы, строит поисковый индекс.
- Поиск. Отвечает на запросы пользователей, генерирует статистику.

Масштабы трагедии

Абсолютные показатели различных экспериментов не совпадают, поэтому необходимо смотреть на отношения.

- Согласно косвенным показателям, количество страниц, доступных для индексирования, в 2005 году составляло 11.5 миллиарда, в 2009 году - 25 миллиардов.
- В соответствии с исследованиями 2001-го года, большая часть документов интернета - 550 миллиардов - не обнаружена поисковыми системами, эту часть называют DeepWeb.
- В 2008 году Google знал 1 триллион уникальных URL-ов.

Так как нет возможности положить в индекс такое количество документов, современная поисковая система производит поиск по десяткам-сотням миллиардов документов.

Uniform Resource Locator

scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]

Один URL можно записать разными способами:

- Схема и имя хоста не чувствительны к регистру.
- Можно не писать стандартный порт.
- Вместо символа можно написать его код через %

GET /index.html HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34
Content-Type: text/html;
charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003
23:11:55 GMT
...

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
    <title>HTML Document</title>
  </head>
  <body>
    <p>
      <b>
        Этот текст будет полужирным,
        <i>а этот — ещё и курсивным</i>
      </b>
    </p>
  </body>
</html>
```

Инвертированный индекс

Определение

Инвертированный индекс - это структура данных, хранящая для каждого слова список документов, в которых это слово встречается.

Постинг лист - вышеупомянутый список документов.

В инвертированном индексе можно ещё хранить и удобно получать доступ к таким данным:

- Свойства самого слова. Например, число его вхождений в корпус.
- Свойства слова и документа. Например, число вхождений слова в документ.

Постинг листы обычно хранят отсортированными по идентификатору документа для ускорения поиска.

Модели поиска

Определение

Булев поиск - первая и самая простая модель информационного поиска. Основывается на выполнении теоретико-множественных операций над списками документов в соответствии с запросом.

- Пусть дан запрос вида $q = (t_1|t_2)\&t_3\ldots$
- На первом шаге необходимо для каждого терма запроса t_i с помощью инвертированного индекса получить список документов, содержащих этот терм.
- На втором шаге необходимо выполнить указанные в запросе операции с полученными множествами документов.

Недостатки булева поиска

- Находит только документы, точно соответствующие запросу. Например, для запроса $q = t_1 \& t_2$ если какой-то документ содержит только терм t_1 , он не найдётся даже если остальные документы не содержат ни одного слова из запроса.
- Не ранжирует результаты поиска.
- Все слова для поиска имеют одинаковую важность, что не соответствует действительности.

Расширенный булев поиск

Недостатки простого булева поиска можно устранить введя в рассмотрение веса термов и модифицировав процедуру поиска:

- $q = \{(t_1, w_{q1}), \dots, (t_n, w_{qn})\}$ - термы запроса со своими весами.
- $d = \{..., (t_1, w_{d1}), \dots, (t_n, w_{dn}), \dots\}$ - вхождения термов запроса в документ, веса соответствующих термов относительно документа.

Замечания:

- Схема выставления весов не является частью модели.
- $w_{\{q|d\}i} \in [0, 1]$
- Документ может и не содержать определённые термы запроса, для таких термов $w_{di} = 0$.

Расширенный булев поиск

Предлагается от простого отношения «слово запроса входит в документ» перейти к учёту весов термов для построения метрики «близости» запроса и документа:

- $sim(d, q = t_1 \& \dots \& t_n) = 1 - (\sum_{i=1}^n w_{qi}^p (1 - w_{di})^p)^{\frac{1}{p}} (\sum_{i=1}^n w_{qi}^p)^{-\frac{1}{p}}$
- $sim(d, q = t_1 | \dots | t_n) = (\sum_{i=1}^n w_{qi}^p w_{di}^p)^{\frac{1}{p}} (\sum_{i=1}^n w_{qi}^p)^{-\frac{1}{p}}$

Замечания:

- p - параметр модели и требует подбора.
- Указанные формулы необходимо рекурсивно применять в соответствии с разбором запроса на элементарные булевы функции, используя в качестве веса для некоторой подформулы значение функции *sim* на ней.

Векторная модель документа

- Документы и запросы представляются в виде разреженных векторов размерности $|T|$ (размер словаря, количество термов):

$$d_i = \{w_{d_i1}, w_{d_i2}, \dots, w_{d_i|T|}\}$$

$$q = \{w_{q1}, w_{q2}, \dots, w_{q|T|}\}$$

- Каждая компонента вектора соответствует появлению определённого терма: если w_{d_ij} отличен от нуля, значит терм t_j встретился в документе d_i .
- Метрика схожести документа и запроса(или другого документа) - косинус угла между их векторными представлениями.

Векторная модель документа

Замечания

- На веса никакие ограничения не накладываются.

Плюсы (относится и к расширенному булевому поиску)

- Учитывает веса слов.
- Допускает отсутствия слов запроса в документе.
- Позволяет ранжировать результаты.

Недостатки (относится и к расширенному булевому поиску)

- Модель подразумевает, что слова появляются в тексте независимо.
- Не учитывается порядок слов.
- Не учитывается смысл документа - если важное слово заменить на синоним, документ может перестать быть релевантным с точки зрения модели.

Расчёт весов термов

Веса термов влияют на разные метрики близости документов.

- Чем больше вес - тем больше вклад в метрику.
- Поэтому хочется давать большой вес «важным» словам.

Как понять, что слово важное?

- Если слово запроса часто встречается в документе, стоит считать его важным.
- Если только это слово не встречается часто во всех документах.

Учтя вышесказанное, возьмём в рассмотрение следующие характеристики:

- Частота терма в документе (tf).
- Доля документов с данным термом (df).

Term Frequency

$f_{t,d}$ - количество вхождений термина t в документ d . $|d|$ - общее количество термов в документе. $tf(t, d)$ - способ придать терму вес относительно данного документа. Возможны варианты:

- $tf(t, d) = \{0, 1\}$ (входит / не входит).
- $tf(t, d) = f_{t,d}$.
- $tf(t, d) = f_{t,d}/|d|$, $tf(t, d) = f_{t,d}/\max_{t' \in d} f_{t',d}$.
- $tf(t, d) = 1 + \log(f_{t,d})$.
- $tf(t, d) = K + (1 - K)f_{t,d}/\max_{t' \in d} f_{t',d}$, $K \in [0, 1]$.

Выбор конкретной схемы зависит от задачи, например, для расширенного булева поиска необходимы веса из $[0, 1]$.

Inverse Document Frequency

$|D|$ - общее количество документов в коллекции D (корпусе). n_t - количество документов, в которых встретился терм t . $idf(t)$ - способ придать вес терму относительно всей коллекции документов, указывающий на количество информации, которое несёт появление термина:

- $idf(t) = \log \frac{|D|}{n_t}$.
- $idf(t) = \log \frac{|D|}{n_t + 1}$.
- $idf(t) = \log \frac{\max_{t' \in d} n_{t'}}{n_t + 1}$.
- $idf(t) = idf(t) / \max_{t' \in d} idf(t')$.

$$tfidf(t, d) = tf(t, d)idf(t)$$

- С одной стороны, компонент $tf(t, d)$ увеличивает вес с увеличением количества вхождений термина в документ.
- С другой стороны, компонент $idf(t)$ стремится к нулю при увеличении доли документов, в которых встретился терм.
- $tfidf(t, d)$ максимален для самого частотного термина t , который встречается только в документе d . Можно считать, что такой терм идеально характеризует свой документ.

Вероятностная модель

В основе модели лежит попытка вероятностно-статистически обосновать понятие релевантности документа запросу и вычислить вероятность того, что пользователь оценит данный документ как релевантный. Воспользовавшись Байесовскими правилами, можно сделать следующие выводы:

- Если $P(R = 1|D) > P(R = 0|D)$, можно считать документ D релевантным.
- $P(R|D) = P(D|R)P(R)/P(D)$.

Все вычисления произведены в условиях наличия некоторого запроса. Проблемой является вычисление вероятностей $P(D|R)$.

Бинарная модель независимости

- С целью сделать $P(D|R)$ вычислимой на практике, предполагается независимость появления термов в документе: $P(D = d|R) = \prod_{i=1}^{|d|} P(t_i|R)$.
- Изначально этот результат использовался для выставления весов термов для векторной модели:

$$w_i = \log \frac{p_i(1 - u_i)}{u_i(1 - p_i)}$$

Где p_i - вероятность встретить терм t_i в релевантном документе, а u_i - в нерелевантном.

Бинарная модель независимости

Теперь проблемой является вычисление $p_i = P(t_i | R = 1)$ и $u_i = P(t_i | R = 0)$. Подход к решению данной проблемы зависит от доступных данных:

- Если на этапе настройки для каждого запроса есть список релевантных документов, можно явно оценить вероятности через частоты с учётом независимости термов.
- Если имеется информация о том, какие документы релевантны некоторым запросам (не известно каким), то можно считать, что распределения термов различаются только между релевантными/нерелевантными документами.
- Если ничего не известно, то можно самим попытаться восстановить множество релевантных документов, например, более строгой моделью поиска.
- ...

$$\text{sim}(d, q) = \sum_{i=0}^{|q|} \text{idf}(t_i) \frac{\text{tf}(t_i, d)(k+1)}{\text{tf}(t_i, d) + k((1-b) + b \frac{|d|}{\text{average}|d_j|, d_j \in D})}$$

- Okapi - поисковая система, созданная в Лондонском городском университете. BM - best match.
- k и b - подбираемые параметры.
- По сей день может использоваться как фактор для более сложных функций ранжирования.

Обобщенная векторная модель

Одним из недостатков векторной модели является предположение о независимости термов. В статье¹ предлагается один из вариантов устранения этого недостатка. Для удобства будем пользоваться обозначениями, введенными авторами:

- $D = \{\bar{d}_\alpha\}, \alpha = 1, \dots, p$ - коллекция документов.
- \bar{q} - запрос
- $\{\bar{t}_i\}, i = 1, \dots, n$ - нормированные, но в общем случае не ортогональные, векторы, соответствующие термам t_i .

¹Wong, S. K. M.; Ziarko, Wojciech; Wong, Patrick C. N. (1985), Generalized vector spaces model in information retrieval

Обобщенная векторная модель

В соответствии с векторной моделью документа и учтя, что векторы термов не ортогональны, а мера схожести - скалярное произведение, можно записать:

- $\bar{d}_\alpha = \sum_{i=1}^n a_{\alpha i} \bar{t}_i$, $\bar{q} = \sum_{j=1}^n q_j \bar{t}_j$ - разложение документа и запроса по векторам $\{\bar{t}_i\}$.
- $\text{sim}(\bar{d}_\alpha, \bar{q}) = \sum_{i=1}^n \sum_{j=1}^n a_{\alpha i} q_j \bar{t}_i \cdot \bar{t}_j$ - их скалярное произведение.

Для вычисления меры схожести нет необходимости знать непосредственное представление для $\{\bar{t}_i\}$, необходимы лишь попарные скалярные произведения. Обратим внимание, что пространство, в котором лежат $\{\bar{t}_i\}$ может быть произвольной размерности.

Обобщенная векторная модель

Отличия обобщённой векторной модели от обыкновенной:

- Отказ от конкретного описания пространства, в котором лежат векторы термов $\{\bar{t}_i\}$. При решении конкретной задачи, такое пространство будет введено явно или неявно, но только для вычисления $\bar{t}_i \cdot \bar{t}_j$.
- Отказ от предположения о независимости термов.

Данные решения усложнили вычисление меры схожести запроса и документа, так как теперь требуются скалярные произведения $\bar{t}_i \cdot \bar{t}_j$, которые раньше выпадали из вычислений из-за независимости термов.

Следующий шаг - привести пример построения скалярного произведения $\bar{t}_i \cdot \bar{t}_j$, обладая только данными из обыкновенной векторной модели.

Обобщенная векторная модель

В качестве $\bar{t}_i \cdot \bar{t}_j$ можно воспользоваться нормализованной поточечной взаимной информацией $NPMI^2$. Если X и Y - дискретные случайные величины, а $P(X = x) = p(x)$, то

$$PMI(x, y) = \log_2 \frac{p(x, y)}{p(x)p(y)} = \log_2 p(x, y) - \log_2 p(x) - \log_2 p(y)$$

$$NPMI(x, y) = PMI(x, y) / (-\log_2 p(x, y))$$

$NPMI(x, y) \in [-1, 1]$ и обладает следующими свойствами:

- $NPMI(x, y) = -1$ для несовместных событий.
- $NPMI(x, y) = 0$ для событий, появляющихся независимо.
- $NPMI(x, y) = 1$ для событий, которые появляются только вместе.

²Bouma, Gerlof (2009). Normalized (Pointwise) Mutual Information in Collocation Extraction

Обобщенная векторная модель

Создатели модели предлагают и свой подход. Начнём с простого примера. Пусть у нас есть коллекция документов D , в которой документы состоят только из двух термов t_1 и t_2 .

- $D = D_{t_1, -t_2} \sqcup D_{t_1, t_2} \sqcup D_{-t_1, t_2}$ - разбиение D на подмножества в соответствии с наличием/отсутствием соответствующего терма.
- Интуиция подсказывает:
 - $|D_{t_1, t_2}|$ должна быть непосредственно связана с $\bar{t}_i \cdot \bar{t}_j$.
 - Вектор \bar{t}_1 должен быть суммой вкладов от $D_{t_1, -t_2}$ и D_{t_1, t_2} (симметрично и для \bar{t}_2). Это предположение является неявным вводом пространства для $\{\bar{t}_i\}$, источником независимости и попыткой получить как можно больше информации от обыкновенной векторной модели.

Обобщенная векторная модель

В соответствии с интуитивными предположениями запишем:

$$\bar{t}_1 = \frac{|D_{t_1, -t_2}| \bar{m}_1 + |D_{t_1, t_2}| \bar{m}_2}{\sqrt{|D_{t_1, -t_2}|^2 + |D_{t_1, t_2}|^2}} \quad \bar{t}_2 = \frac{|D_{t_1, t_2}| \bar{m}_2 + |D_{-t_1, t_2}| \bar{m}_3}{\sqrt{|D_{t_1, t_2}|^2 + |D_{-t_1, t_2}|^2}}$$

$$\bar{t}_1 \cdot \bar{t}_2 = \frac{|D_{t_1, t_2}|^2}{\sqrt{|D_{t_1, -t_2}|^2 + |D_{t_1, t_2}|^2} \sqrt{|D_{t_1, t_2}|^2 + |D_{-t_1, t_2}|^2}}$$

Где $\{\bar{m}_1, \bar{m}_2, \bar{m}_3\}$ - некоторый ортонормированный базис.
Каждый вектор \bar{m}_i отвечает некоторому максимальному по размеру подмножеству документов с одинаковой «маской» вхождения/невхождения термов.

Обобщенная векторная модель

Усложним ситуацию. Пусть теперь:

$$D = D_{t_1, t_2, t_3} \sqcup D_{-t_1, t_2, t_3} \sqcup \dots \sqcup D_{-t_1, t_2, -t_3} \sqcup D_{-t_1, -t_2, t_3}$$

Теперь у t_1 (как и у остальных) больше способов войти в документ относительно наличия остальных термов:

$$\bar{t}_1 = \frac{1}{N_1} (|D_{t_1, t_2, t_3}| \bar{m}_1 + |D_{t_1, -t_2, t_3}| \bar{m}_3 + |D_{t_1, t_2, -t_3}| \bar{m}_4 + |D_{t_1, -t_2, -t_3}| \bar{m}_5)$$

$$N_1 = \sqrt{|D_{t_1, t_2, t_3}|^2 + |D_{t_1, -t_2, t_3}|^2 + |D_{t_1, t_2, -t_3}|^2 + |D_{t_1, -t_2, -t_3}|^2}$$

...

Выражение для скалярного произведения примет вид:

$$\bar{t}_1 \cdot \bar{t}_2 = \frac{1}{N_1 N_2} (|D_{t_1, t_2, t_3}|^2 + |D_{t_1, t_2, -t_3}|^2)$$

Обобщенная векторная модель

Для упрощения записи формул случая общего вида воспользуемся обозначениями из булевой алгебры.

- Пусть $m_k = t_1^{\delta_1} \wedge \dots \wedge t_n^{\delta_n}$ - некоторый элемент булевой алгебры, отвечающий вектору \bar{m}_k . Где

$$t_i^{\delta_i} = \begin{cases} t_i & \text{если } \delta_i = 1 \\ \neg t_i & \text{иначе} \end{cases}$$

обозначает наличие или отсутствие соответствующего терма в документе.

- Каждый элемент t_i может быть представлен в виде $t_i = m_{i_1} \vee \dots \vee m_{i_r}$, где операция производится над всеми такими m_{i_j} , для которых $m_{i_j} \vee t_i = t_i$.
- В соответствии с предыдущим пунктом: $\bar{t}_i = \sum_{j=1}^r c_j(t_i) \bar{m}_{i_j}$

Обобщенная векторная модель

Стоит сделать несколько замечаний:

- $\{\bar{m}_1, \dots, \bar{m}_{2^n}\}$ - как было сказано выше, произвольный ортонормированный базис. В явном виде эти векторы при вычислениях никогда не появляются.
- Как вычислять $c_j(t_i)$? Возможны несколько вариантов:
 - Положить $c_j(t_i) = 1$. Самый простой случай, соответствует обычному булеву поиску (не в прямом смысле).
 - Вычислять $c_j(t_i)$ как было предложено при рассмотрении простых примеров.
 - Попытаться обобщить предыдущий вариант и воспользоваться изначальным векторным представлением документов.

Обобщенная векторная модель

Воспользуемся весами термов $a_{\alpha i}$ из обыкновенной векторной модели для подсчёта $c_j(t_i)$.

- Пусть для каждого $m_k = t_1^{\delta_1} \wedge \dots \wedge t_n^{\delta_n}$:

$$D(m_k) = D_{t_1}^{\delta_1} \cap \dots \cap D_{t_n}^{\delta_n}$$

$$D_{t_i}^{\delta_i} = \begin{cases} D_{t_i} & \text{если } \delta_i = 1 \\ D_{-t_i} & \text{иначе} \end{cases}$$

- Тогда $c_j(t_i) = \sum_{\{\alpha: d_\alpha \in D(m_{ij})\}} a_{\alpha i}$ и если необходимо занормировать \bar{t}_i :

$$\bar{t}_i = \frac{1}{N_i} \sum_{j=1}^r c_j(t_i) \bar{m}_{ij} \quad N_i = \sqrt{\sum_{j=1}^r c_j^2(t_i)}$$

Обобщенная векторная модель

Достоинства обобщённой векторной модели:

- Отказ от изначально неверного предположения о независимости термов, что даёт более точную меру схожести.

Недостатки:

- Сложна для вычисления, требует хранить матрицу скалярных произведений.
- Не учитывает порядок термов (можно исправить с помощью n -граммного индекса).

Latent Semantic Indexing

Как уже было сказано выше, проблемами векторной модели являются:

- Предположение о независимости термов.
- Синонимы. Одинаковый смысл может быть передан разными словами.
- Многозначные слова, имеющие разный смысл в зависимости от контекста.

LSI³ была разработана с учётом всех этих недостатков.

³Deerwester, S., et al, Improving Information Retrieval with Latent Semantic Indexing. (1988)

Latent Semantic Indexing

- Идея (Deerwester et al):
“We would like a representation in which a set of terms, which by itself is incomplete and unreliable evidence of the relevance of a given document, is replaced by some other set of entities which are more reliable indicants. We take advantage of the implicit higher-order (or latent) structure in the association of terms and documents to reveal such relationships.”
- Будем полагаться на принцип: чем чаще два слова встречаются в похожих контекстах, тем ближе они по смыслу.

Latent Semantic Indexing

Важно учитывать, что подразумевается под «контекстом»:

- Word2Vec - несколько слов перед и после текущей позиции.
- LSI - факт появления в одном документе.

LSI заключается в переходе из исходного пространства классической векторной модели в пространство меньшей размерности. Данный переход можно осуществить с помощью *SVD*. Для вычислений по модели LSI достаточно матрицы *терм-документ*.

Latent Semantic Indexing

Последовательность шагов:

- Построить матрицу *терм-документ* $A = \{a_{ij} = tf_{ij}\}$ (где каждому терму соответствует строка, а документу - столбец), заполненную частотами термов.
- Отнормировать матрицу A следующим образом (можно и другими способами):

$$\hat{a}_{ij} = g_i \log(a_{ij} + 1), \quad g_i = \sum_{j=1}^{|D|} \frac{p_{ij} \log p_{ij}}{\log |D|}, \quad p_{ij} = \frac{tf_{ij}}{\sum_{k=1}^{|D|} tf_{ik}}$$

(чтобы понять, откуда столько логарифмов, следует обратиться к преобразованию Бокса-Кокса. *SVD* предполагает нормальность данных, а частоты слов далеко не нормальны - они подчиняются закону Ципфа)

Latent Semantic Indexing

Последовательность шагов:

- С помощью SVD (Truncated SVD) построить разложение $\hat{A} \approx TSD^T$ с r наибольшими сингулярными числами. Представление документа во внутреннем r -мерном пространстве и есть искомое представление. Каждая компонента в нём - соответствие некоторой «тематике».
- Для необходимых документов выполнить преобразование $d_{LSI} = d^T TS^{-1}$, где d - столбец \hat{A} или соответствующим образом преобразованный запрос в формате векторной модели.
- Выполнить сравнения с помощью скалярного произведения в r -мерном пространстве.

Latent Semantic Indexing

Достоинства модели:

- Учитывает зависимость появления термов, наличие синонимии и нескольких смыслов у слов.

Недостатки:

- Вычислительная сложность при построении модели.
- Вычислительная сложность при поиске - необходимо сравнить запрос с каждым документом в коллекции.

Языковые модели, лингвистика и NLP

Определение

Языковая модель - это вероятностное распределение на последовательностях термов.

В большинстве задач информационного поиска необходимо вычислять вероятность появления заданной последовательности слов или символов. Мы уже убедились, что это необходимо как минимум для построения адекватной меры схожести запросов и документов.

Но необходимость в языковой модели может появиться уже раньше: в языке может не быть выделенной границы между словами, или, например, могут опускаться гласные.

Другие применения языковой модели:

- Порождение текста - генеративная модель языка.
- Распознавание речи. Необходимо определить, какая из интерпретаций фразы наиболее вероятна.
- Категоризация текстов. Модель может быть построена не по всему языку, а например, только по научным статьям.

Рассмотрим некоторые простые закономерности, которые могут быть полезны при работе с текстом.

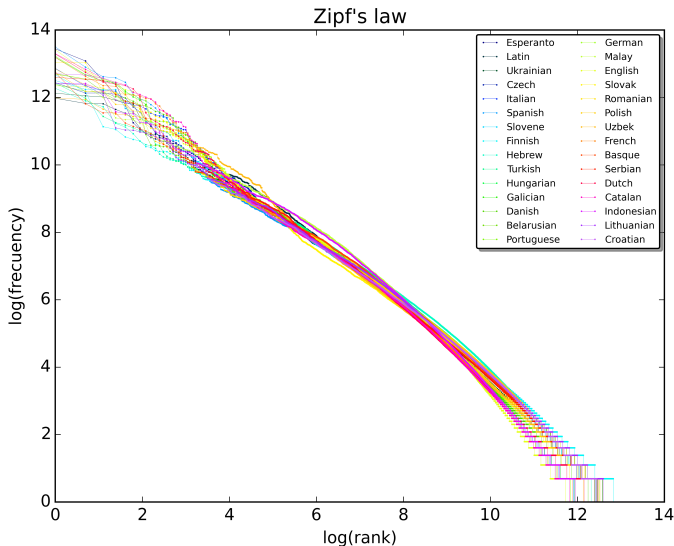
Простые закономерности. Zipf Law

На практике часто оказывается, что некоторые характеристики исследуемых объектов, такие как частота (появления) или размер (группы), подчиняются одному закону - закону Ципфа, который связывает вероятность появления объекта с его рангом k в списке, отсортированном по некоторому параметру:

$$f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^N 1/n^s}$$

Здесь s - параметр модели. Простыми словами: количество объектов, находящихся в списке на позиции k , обратно пропорционально k в некоторой степени s .

Простые закономерности. Zipf Law



Простые закономерности. Zipf Law

Закону Ципфа подчиняются:

- Частоты слов в большинстве языков (даже искусственных).
- Размеры городов. Размеры корпораций.
- Прибыль, зарплаты.
- Аудитории телеканалов.
- Размеры групп похожих документов в интернете.

Простые закономерности. Hears' Law

Другой полезный закон - Hears' law, связан с увеличением словаря при добавлении новых документов. Его также называют Herdan's law. Закон выражает размер словаря как функцию от количества документов:

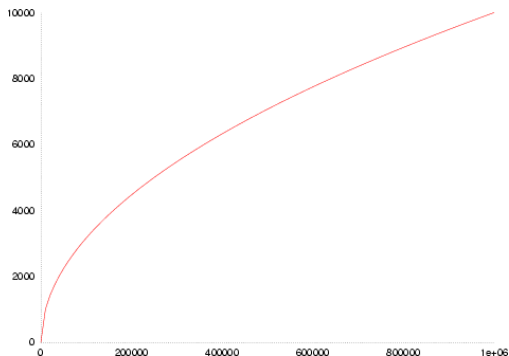
$$|T| = K|D|^{\beta}$$

Здесь β - параметр модели.

Также, как и закон Ципфа, данный закон применим не только к словарю термов коллекции документов.

Простые закономерности. Zipf Law

Примерная зависимость количества слов от размера текста:



Униграммная модель языка

Самая простая модель, для вычисления вероятности последовательности термов использует только маргинальные вероятности:

$$P(t_1 t_2 t_3) = P(t_1)P(t_2|t_1)P(t_3|t_1 t_2)$$

$$P_{1-gramm}(t_1 t_2 t_3) \approx P(t_1)P(t_2)P(t_3)$$

Знак \approx в данном случае не означает примерное совпадение, он лишь указывает на способ приближения. Данная модель применялась нами ранее в простых моделях поиска.

N-граммная модель языка

Недостатки униграммной модели очевидны. Для построения более правдоподобной модели необходимо учитывать контекст:

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_1 t_2)P(t_4|t_1 t_2 t_3)$$

$$P_{2\text{-gramm}}(t_1 t_2 t_3 t_4) \approx P(t_1)P(t_2|t_1)P(t_3|t_2)P(t_4|t_3)$$

$$P_{3\text{-gramm}}(t_1 t_2 t_3 t_4) \approx P(t_1)P(t_2|t_1)P(t_3|t_1 t_2)P(t_4|t_2 t_3)$$

$$P_{n\text{-gramm}}(t_1 \dots t_m) \approx \prod_{i=1}^m P(t_i | t_{i-(n-1)} \dots t_{i-1})$$

Здесь $P(t_i | t_{i-1})$ означает вероятность встретить терм t_i после t_{i-1} . Следует заметить, что отдельный интерес могут представлять и сами вероятности $P(t_i | t_{i-(n-1)} \dots t_{i-1})$ - вероятности встретить слово в текущем контексте.

N-граммная модель языка. Skip-gram

Для некоторых задач, в которых необходимо, например, учесть и левый и правый контекст слова или снизить влияние перестановок рядом стоящих слов, можно воспользоваться подходом k -skip- n -gram.

Определение

k -skip- n -gram - подпоследовательность исходного текста, обладающая длиной k , в которой каждый терм находится в тексте на расстоянии, не большем, чем n .

N-граммная модель языка. Сглаживание

Для работы с n-граммными моделями (и другими, основанными на умножении вероятностей) необходимо знать условные вероятности появления термов, например, для биграммной модели: $P(t_i|t_{i-1}) = c(t_{i-1}t_i) / \sum_j c(t_{i-1}t_j)$.

Если в тексте, для которого вычисляется вероятность, встречается новое для модели слово, итоговая вероятность $P_{n-gramm}(t_1...t_m)$ будет равна нулю. Это, обычно, не то, что требуется.

Самая простая идея, «add-one smoothing»:

$$P(t_i|t_{i-1}) = \frac{1 + c(t_{i-1}t_i)}{|T| + \sum_j c(t_{i-1}t_j)}$$

В дальнейшем будем рассматривать методы сглаживания на примере биграммной модели.

Сглаживание: Additive

Будем считать, что мы «видели» каждую n -грамму на δ раз больше, чем на самом деле:

$$P_{add}(t_i | t_{i-1}) = \frac{\delta + c(t_{i-1}t_i)}{\delta |T| + \sum_j c(t_{i-1}t_j)}$$

Обычно выбирают $0 < \delta \leq 1$.

Это один из самых простых методов, но он достаточно не точный.

Сглаживание: Good-Turing estimation

- Предположим, некоторая n -грамма α появляется в корпусе с некоторой неизвестной вероятностью p .
- Допустим, что все вхождения α в корпус независимы, а количество вхождений подчинено биномиальному распределению:

$$P(c(\alpha) = r) = C_N^r p^r (1 - p)^{N-r}$$

- Для построения модели нам необходимо найти ожидаемое количество вхождений:

$$E_N[c^*(\alpha)] = \sum_{r=0}^N r C_N^r p^r (1 - p)^{N-r}$$

- Но нам не известна p .

Сглаживание: Good-Turing estimation

- Пусть n_r - количество n -грамм, которые встретились ровно r раз.
- Допустим, мы имеем s различных n -грамм $\alpha_1, \dots, \alpha_s$, которые встречаются с вероятностями p_1, \dots, p_s соответственно.
- Тогда

$$E_N[n_r] = \sum_{i=1}^s P(c(\alpha_i = r)) = \sum_{i=1}^s C_N^r p_i^r (1 - p_i)^{N-r}$$

Сглаживание: Good-Turing estimation

- Нам не известны p_i , но у нас есть формула для $E_N[n_r]$, мы можем подсчитать n_r по корпусу и надеяться, что для небольших r выполнено $E_N[n_r] \approx n_r$.
- Так как нам известно конкретное количество вхождений r :

$$E_N[c^*(\alpha) | c(\alpha) = r] = N \sum_{i=1}^s p_i \frac{P(c(\alpha_i) = r)}{\sum_{j=1}^s P(c(\alpha_j) = r)}$$

так как каждая из s n -грамм может встречаться r раз и

$$P(\alpha = \alpha_i | c(\alpha) = r) = \frac{P(c(\alpha_i) = r)}{\sum_{j=1}^s P(c(\alpha_j) = r)}$$

Сглаживание: Good-Turing estimation

$$E_N[c^*(\alpha)|c(\alpha) = r] = \frac{N \sum_{i=1}^s p_i P(c(\alpha_i) = r)}{\sum_{j=1}^s P(c(\alpha_j) = r)}$$

- Знаменатель - $E_N[n_r]$ по определению.
- Раскроем скобки в числителе:

$$\begin{aligned} N \sum_{i=1}^s p_i P(c(\alpha_i) = r) &= (r+1) \frac{N}{N+1} \sum_{i=1}^s C_{N+1}^r p_i^{r+1} (1-p_i)^{N-r} = \\ &= (r+1) \frac{N}{N+1} E_{N+1}[n_{r+1}] \approx (r+1) E_{N+1}[n_{r+1}] \end{aligned}$$

Сглаживание: Good-Turing estimation

- Подставив всё в исходную формулу, получим:

$$r^* = E_N[c^*(\alpha) | c(\alpha) = r] = (r + 1) \frac{E_{N+1}[n_{r+1}]}{E[n_r]}$$

- Можно сделать следующее допущение:

$$r^* = (r + 1) \frac{n_{r+1}}{n_r}$$

- Как интерпретировать этот результат?

Сглаживание: Good-Turing estimation

- Интерпретация: давайте «перенесём» часть вероятности с n -грамм, которые встретились нам $r + 1$ раз, на n -граммы, которые встретились r раз.
- Как следствие, перенесём вероятность с n -грамм, которые встретились по одному разу, на ни разу не встречающиеся n -граммы.
- Пусть n_r - количество n -грамм, встретившихся ровно r раз. Для каждого r необходимо вычислить

$$r^* = (r + 1) \frac{n_{r+1}}{n_r}, \quad n_0 = N$$

- Тогда при $N = \sum_{r=0}^{\infty} r^* n_r = \sum_{r=1}^{\infty} r n_r$ будем иметь:

$$P_{GT}(t_i | t_{i-1} : c(t_{i-1} t_i) = r) = r^* / N$$

Сглаживание: Good-Turing estimation

Данный подход обладает рядом недостатков и сталкивается с некоторыми проблемами:

- Для больших r часто встречается ситуация $n_r = 0$.
- Для больших r , даже при отсутствии «дыр» в частотах, значения n_r достаточно зашумлены, ведь правильнее писать:

$$r^* = (r + 1) \frac{E[n_{r+1}]}{E[n_r]}$$

- Но как вычислить математические ожидания?

Данные проблемы делают метод Good-Turing estimation неудобным для применения «как есть». Но он является фундаментом для построения других методов.

Сглаживание: Jelinek-Mercer interpolation

Допустим, $c(\text{мама мыла}) = 0$ и $c(\text{мама шыла}) = 0$ тогда:

- Рассмотренные до этого момента методы дадут одинаковый результат:

$$P(\text{мыла}|\text{мама}) = P(\text{шыла}|\text{мама})$$

- Что выглядит неправдоподобно, так как, очевидно, что:

$$P(\text{мыла}|\text{мама}) > P(\text{шыла}|\text{мама})$$

- Необходимо «смешать» биграммную и униграммную модели.

Сглаживание: Jelinek-Mercer interpolation

- Смесь моделей можно записать в виде:

$$P_{JM}(t_i|t_{i-1}) = \lambda_{t_{i-1}}P(t_i|t_{i-1}) + (1 - \lambda_{t_{i-1}})P(t_i)$$

- $\lambda_{t_{i-1}}$ можно подобрать ЕМ-алгоритмом, или, например, задать одинаковыми исходя из представлений о языковой модели. Но оптимальные значения должны быть связаны с контекстом t_{i-1} : для частого контекста - большие значения $\lambda_{t_{i-1}}$.
- Естественным рекурсивным образом распространяется на остальные n -граммные модели для $n \geq 2$.

Сглаживание: Katz smoothing

- Как и в методе Good-Turing, предлагается искусственно снизить частоты биграмм, которые встретились $r \geq 1$ раз, с помощью понижающего коэффициента d_r :

$$c_{katz}(t_{i-1}t_i) = d_{c(t_{i-1}t_i)}c(t_{i-1}t_i)$$

- Полученную массу необходимо распределить между биграммами, встретившимися 0 раз, в соответствии с языковой моделью предыдущего порядка (униграммной моделью):

$$c_{katz}(t_{i-1}t_i) = \alpha(t_{i-1})c(t_i)$$

- Построенная модель будет иметь вид:

$$P_{katz}(t_i|t_{i-1}) = \frac{c_{katz}(t_{i-1}t_i)}{\sum_j c_{katz}(t_{i-1}t_j)}$$

- $\alpha(t_{i-1})$ вычисляются исходя из условия $\sum_i P_{katz}(t_i|t_{i-1}) = 1$:

$$\alpha(t_{i-1}) = \frac{1 - \sum_{i:c(t_{i-1}t_i)>0} P_{katz}(t_i|t_{i-1})}{\sum_{j:c(t_{i-1}t_j)=0} P(t_j)}$$

Сглаживание: Katz smoothing

Для вычисления d_r Katz предлагает следующий подход:

- Если $r > k$ (обычно $k = 5$), то будем считать, что сглаживание не требуется: $d_r = 1$.
- В остальных случаях потребуем соответствия модели Good-Turing:
 - $1 - d_r = \mu(1 - r^*/r)$ - «унесённая» масса пропорциональна соответствующей величине в Good-Turing.
 - $\sum_{r=1}^k n_r(1 - d_r)r = n_1$ - суммарная масса, перенесённая на неизвестные биграммы, такая же, как и в Good-Turing.

- Подставив всё в исходную формулу, можно получить:

$$d_r = \frac{\frac{r^*}{r} - (k+1)\frac{n_{k+1}}{n_1}}{1 - (k+1)\frac{n_{k+1}}{n_1}}$$

- Аналогично Jelinek-Mercer, рекурсивно распространяется на n-граммные модели более высоких порядков.

Сглаживание: Witten-Bell smoothing

- Выпишем выражение для сглаживания Jelinek-Mercer:

$$P_{JM}(t_i|t_{i-1}) = \lambda_{t_{i-1}}P(t_i|t_{i-1}) + (1 - \lambda_{t_{i-1}})P(t_i)$$

- Идея: $\lambda_{t_{i-1}}$ можно интерпретировать как вероятность использования модели более высокого порядка.
- Значит $1 - \lambda_{t_{i-1}}$ - вероятность встретить терм t_i в тестовых данных после термина t_{i-1} , при условии, что он не встречался в обучающей выборке в таком контексте.
- Эту вероятность можно оценить с помощью количества уникальных термов, встречавшихся после t_{i-1} .

Сглаживание: Witten-Bell smoothing

- Вычислим количество уникальных термов, встречавшихся после t_{i-1} :

$$N_{1+}(t_{i-1}\bullet) = |\{t_i : c(t_{i-1}t_i) > 0\}|$$

- Тогда выражение для $\lambda_{t_{i-1}}$ примет вид:

$$1 - \lambda_{t_{i-1}} = \frac{N_{1+}(t_{i-1}\bullet)}{N_{1+}(t_{i-1}\bullet) + \sum_j c(t_{i-1}t_j)}$$

Сглаживание: Absolute discounting

- Основано на общей с Jelinek-Mercer идее интерполяции моделей разных порядков.
- Но вместо вычисления $\lambda_{t_{i-1}}$ предлагается вычесть фиксированное $\delta \in [0, 1]$ из всех не нулевых количеств:

$$P_{abs}(t_i|t_{i-1}) = \frac{\max(c(t_{i-1}t_i) - \delta, 0)}{\sum_j c(t_{i-1}t_j)} + \frac{\delta N_{1+}(t_{i-1}\bullet)}{\sum_j c(t_{i-1}t_j)} P(t_i)$$

Сглаживание: Kneser-Ney smoothing

- Основано на модели Absolute discounting с более точным учётом модели низкого порядка: влияние модели низкого порядка должно быть тем больше, чем меньше количество вхождений в модели более высокого порядка.
- Введём обозначения:

$$N_{1+}(\bullet t_i) = |\{t_{i-1} : c(t_{i-1}t_i) > 0\}|$$

$$N_{1+}(\bullet\bullet) = \sum_i N_{1+}(\bullet t_i)$$

$$P_{KN}(t_i) = \frac{N_{1+}(\bullet t_i)}{N_{1+}(\bullet\bullet)}$$

Сглаживание: Kneser-Ney smoothing

Сведём всё воедино:

$$P_{KN}(t_i|t_{i-1}) = \frac{\max(c(t_{i-1}t_i) - \delta, 0)}{\sum_j c(t_{i-1}t_j)} + \frac{\delta N_{1+}(t_{i-1}\bullet)}{\sum_j c(t_{i-1}t_j)} P_{KN}(t_i)$$

Модификация данного подхода, заключающаяся в подборе отдельного δ для каждого n , является одним из лучших методов сглаживания.

Поисковый робот

- Система для массового скачивания и обработки документов.
- Возможны «pull» и «push» модели:
 - В «push» модели поставщик сам отправляет данные в поисковик.
 - В «pull» модели поисковик находит и скачивает документы для индексации.

«push» модель не прижилась по нескольким причинам, в том числе из-за сложности.

- Базовый алгоритм: начав с некоторого начального множества URL-ов (seed-ы) в очереди, скачать все доступные документы, распарсить, найденные ссылки положить в очередь и перейти к начальному шагу.

- Масштаб и масштабируемость. Интернет содержит огромное количество страниц, растущее с каждым днём. Многие страницы достаточно часто обновляются.
- Выбор контента. Нет возможности ни скачать весь интернет, ни сохранить его на диске.
- Вежливость. Робот не должен создавать излишней нагрузки на обходимые сайты.
- Интеллектуальность. Необходимо избегать ловушек - специально организованных множеств страниц, по которым робот может ходить бесконечно.

- Свежесть и возраст vs полнота. Небольшое подмножество документов обновляется очень часто, например, новостные сайты. Другие обновляются редко, но их на много больше.
- Хранение скачанного контента. Большому роботу требуется специализированное хранилище данных.
- Дублирующийся контент.
- Deep Web. Не весь полезный контент доступен по прямым ссылкам.

Взаимодействие с администратором ресурса

- RSS. Упрощает поиск новых документов на сайте.
- Sitemap. Содержит следующую информацию:
 - Список документов на сайте.
 - Время последнего обновления документов.
 - Примерную частоту обновлений.
- robots.txt. Содержит указания для робота относительно того, какие разделы сайта разрешены или запрещены к индексации.

Нормализация URL

Нормализации, сохраняющие исходное написание:

- Конвертация в нижний регистр схемы и хоста.
- Перевод в верхний регистр управляющих конструкций.
- Перекодировка управляющих конструкций в явные символы.
- Удаление порта по умолчанию.

Нормализация с частичным сохранением исходного написания:

- Добавление конечного слеша.
- Удаление сегментов-точек.

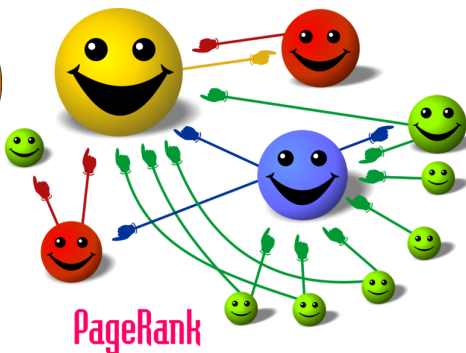
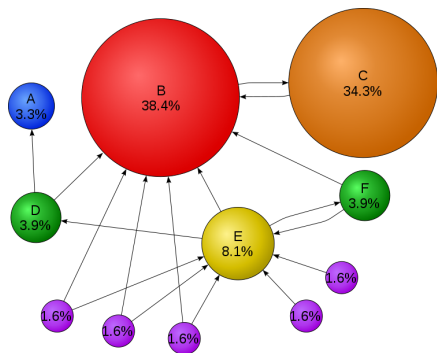
Нормализация URL

Нормализации, изменяющие написание:

- Удаление головного индекса.
- Удаление фрагментов.
- Замена IP адреса именем домана.
- Сокращение идентификаторов протоколов.
- Удаление дублированных слешей.
- Удаление или добавление «www» как элемента верхнего доменного уровня.
- Сортировка параметров запросов.
- Удаление неиспользуемых переменных в запросе.
- Удаление параметров запроса по умолчанию.
- Удаление «?» при пустом запросе.

Page Rank

Характеристика «важности» веб-страницы, предложенная⁴ С. Брином и Л. Пейджем, основанная на предположении, что «хорошие» страницы должны ссылаться на хорошие.



⁴Brin, S.; Page, L. (1998). "The anatomy of a large-scale hypertextual Web search engine"

Page Rank задаёт вероятностное распределение на страницах. Соответствующие значения можно интерпретировать как вероятность того, что случайно блуждающий по ссылкам пользователь окажется на текущей странице.

$$PR(d_i) = \frac{1 - \delta}{N} + \delta \sum_{d_j \in M(d_i)} \frac{PR(d_j)}{|L(d_j)|}$$

где d_1, \dots, d_N - рассматриваемые страницы, $M(d_i)$ - множество ссылок, входящих в d_i , $L(d_j)$ - множество исходящих ссылок с d_j .

- Итеративное. Можно задаться начальным приближением $PR_0 = \frac{1}{N}\mathbf{1}$ и действовать итеративно:

$$PR_{t+1} = \frac{1-\delta}{N}\mathbf{1} + \delta MR_t, \quad M_{ij} = \begin{cases} 1/|L(p_j)| & , d_j \rightarrow d_i \\ 0 & \end{cases}$$

- Точное. Алгебраическое решение исходного уравнения выглядит следующим образом:

$$PR = \frac{1-\delta}{N}(I - \delta M)^{-1}$$

Допустим, что M - стохастическая матрица. Тогда, если E - матрица из единиц:

$$PR = (\delta M + \frac{1 - \delta}{N} E) PR = \hat{M} PR$$

PR - собственный вектор матрицы \hat{M} , соответствующий максимальному собственному числу (единице).

Power Method $PR_{t+1} = \hat{M} PR_t$ сходится как раз к такому собственному вектору.

Hyperlink-Induced Topic Search (HITS)

Вычисляет для документа два показателя: «хабовость» и «авторитетность».

- Чем больше данная страница ссылается на авторитетные страницы, тем выше её «хабовость».
- Чем больше хабовых страниц ссылаются на данную, тем выше её «авторитетность».

Аналогично Page Rank, необходимо итеративно обновлять значения:

$$auth(d_i) = \sum_{d_j \in M(d_i)} hub(d_j) \quad hub(d_i) = \sum_{d_j \in L(d_i)} auth(d_j)$$

HITS. Отличия от Page Rank

- Редко использовался поисковыми системами.
- Выполняется во время запроса над небольшим количеством документов.
- Вычисляет два показателя.

Сходимость Power Method

- Требования:
 - У матрицы A есть одно собственное значение $\lambda_1 > 0$, строго большее по модулю, чем остальные.
 - x_0 не ортогонален v_1 - собственному вектору, соответствующему λ_1 .
- Итеративный процесс вида $x_{k+1} = \frac{Ax_k}{\|Ax_k\|}$.
- На каждой итерации происходит одно умножение матрицы на вектор. Никакие разложения матрицы не нужны.
- Процесс сходится к таким x и λ , что $Ax = \lambda x$, $|\lambda| = |\lambda_1|$. Почему?

Сходимость Power Method

- Разложим A в жорданову нормальную форму $A = VJV^{-1}$, где первый столбец V - v_1 , а первая жорданова клетка в J имеет размер 1×1 и равна $[\lambda_1]$. Существование именно такого разложения следует из исходных требований.
- Представим x_0 в виде $x_0 = c_1 v_1 + \dots + c_n v_n$, где $c_1 \neq 0$. Тогда:

$$x_{k+1} = \frac{Ax_k}{\|Ax_k\|} = \frac{A^{k+1}x_0}{\|A^{k+1}x_0\|}$$

Сходимость Power Method

Упростим степень матрицы:

$$x_k = \frac{A^k x_0}{\|A^k x_0\|} = \frac{(VJV^{-1})^k x_0}{\|(VJV^{-1})^k x_0\|} = \frac{VJ^k V^{-1} x_0}{\|VJ^k V^{-1} x_0\|}$$

Внесём V^{-1} в скобки:

$$x_k = \frac{VJ^k V^{-1}(c_1 v_1 + \dots + c_n v_n)}{\|VJ^k V^{-1}(c_1 v_1 + \dots + c_n v_n)\|} = \frac{VJ^k(c_1 e_1 + \dots + c_n e_n)}{\|VJ^k(c_1 e_1 + \dots + c_n e_n)\|}$$

Сходимость Power Method

Вынесем первое слагаемое из под скобок:

$$x_k = \frac{c_1 VJ^k e_1 + VJ^k(\sum_{i=2}^n c_i e_i)}{\|c_1 VJ^k e_1 + VJ^k(\sum_{i=2}^n c_i e_i)\|} = \frac{c_1 \lambda_1^k v_1 + VJ^k(\sum_{i=2}^n c_i e_i)}{\|c_1 \lambda_1^k v_1 + VJ^k(\sum_{i=2}^n c_i e_i)\|}$$

Вынесем за скобку $c_1 \lambda_1^k$:

$$x_k = \left(\frac{\lambda_1}{|\lambda_1|} \right)^k \frac{c_1}{|c_1|} \frac{v_1 + \frac{1}{c_1} V(\frac{1}{\lambda_1} J)^k(\sum_{i=2}^n c_i e_i)}{\|v_1 + \frac{1}{c_1} V(\frac{1}{\lambda_1} J)^k(\sum_{i=2}^n c_i e_i)\|}$$

Сходимость Power Method

Пусть J_1, \dots, J_m - жордановы клетки из матрицы J , тогда:

$$\left(\frac{1}{\lambda_1} J_1\right)^k \rightarrow [1], \text{ при } k \rightarrow \infty$$

$$\left(\frac{1}{\lambda_1} J_i\right)^k \rightarrow \mathbf{0}, \text{ при } k \rightarrow \infty, \text{ для } i \geq 2$$

Так как J составлена из блоков J_i :

$$\left(\frac{1}{\lambda_1} J\right)^k \rightarrow \text{diag}\{1, 0, \dots, 0\}, \text{ при } k \rightarrow \infty$$

Сходимость Power Method

Соответственно, можно записать:

$$\frac{1}{c_1} V \left(\frac{1}{\lambda_1} J \right)^k \left(\sum_{i=2}^n c_i e_i \right) \rightarrow 0, \text{ при } k \rightarrow \infty$$

$$x_k \rightarrow \frac{c_1}{|c_1|} v_1, \text{ при } k \rightarrow \infty$$

При вычислении Page Rank таким методом, теорема Перрона-Фробениуса и теория стохастических матриц помогают нам следующими фактами:

- $\lambda_1 = 1$.
- Все компоненты A , x_k , v_1 положительны, а следовательно и $c_1 > 0$.

Power Method

- Может быть использован для больших матриц, так как не требует сложных разложений.
- Для многомиллионных ссылочных графов сходится за несколько десятков итераций. Но на практике нет необходимости в такой точности и достаточно меньшего количества итераций.
- Применив преобразование $\tilde{A} = A - \lambda I$ и воспользовавшись методом повторно, можно найти второй собственный вектор...
- На матрицах со значением $|\lambda_2/\lambda_1|$, близким к 1, может долго сходиться (долго не сходиться).

Обнаружение дубликатов

Зачем находить дубли?

- Экономия процессорного времени при сложных операциях с содержимым документа.
- Экономия сетевых ресурсов при передаче контента.
- Экономия дискового пространства.
- Перенос полезного сигнала с дублей на главный документ.
- Пользователи не любят дублирующийся контент в выдаче.

- Точные дубли.
- Нечёткие дубли (полудубли).
- Зеркала.
- Редиректы (3XX, JS, meta-refresh).
- `rel="canonical"`.
- Незначащие параметры.
- DUST.

- Автоматическое перенаправление браузера с одного URLa на другой.
- Могут быть организованы как:
 - Код 301 - постоянное перенаправление.
 - Код 302/303/307 - временное перенаправление.
 - Перенаправление meta `http-equiv="refresh"`.
 - Перенаправление с помощью JavaScript.

Зеркала, rel="canonical"

- Зеркала:
 - Разные хосты с одинаковым контентом.
 - www/без www, http/https, бекап, переезд...
 - Для обнаружения необходимо одновременно скачивать одинаковые пути с разных сайтов.
- rel="canonical":
 - Используется вебмастерами.
 - Способ явно указать каноническую версию документа в html.

Different URLs with Similar Text (DUST)

- Урлы, преобразуемые друг в друга по определённым правилам, специфичным для хоста:
 - <http://google.com/news>
 - <http://news.google.com>
- По коллекции документов набирается статистика для p и s по дубликатам вида:

$$url_1 = p\alpha s \quad url_2 = p\beta s$$

Контентные дубли

- Точные дубли:
 - Бинарное совпадение индексируемого содержимого документов.
 - Может быть легко организовано в виде сравнения сигнатур (CRC, MD5, SHA).
 - Легко находить группы одинаковых документов.
 - Процедура взятия сигнатуры обычно не устойчива к малым изменениям документа.
- Полудубли:
 - Практически полное совпадение индексируемого содержимого.
 - Что значит «практически»? Нужна метрика схожести.
 - Не все данные из документа необходимо учитывать.

- По коллекции документов строится словарь всех термов L .
- Производится фильтрация словаря: остаются только «важные» слова. Важность определяется экспертами или, например, на основе *tfidf*: удаляются слова с высоким и низким значением *tfidf*.
- Сигнатурой документа объявляется хэш от его пересечения со словарём. Для сравнения документов теперь необходимо сравнить хэши.
- Вместо однократной фильтрации можно сгенерировать несколько словарей, случайно выкинув 30 — 40% термов из исходного словаря.
- Обладая несколькими сигнатурами, можно «придумать» много метрик схожести.

Шинглы (Shingles)

- В контексте сравнения текстовых документов, шингл - последовательность термов заданной длины (N-грамма).
- Хранить N-граммы в исходном виде нет необходимости - можно от каждой взять небольшой хэш (CRC64).
- Документ представляется множеством шинглов. Схожесть документов, например - мера Жаккарда:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Текущая постановка по сложности сопоставима с прямым сравнением документов, так как количество шинглов не на много меньше количества термов в документе.

Шинглы, Min-Hash

- Предположим, что для документов рассчитаны наборы шинглов ($S_D \subseteq \{0, \dots, n - 1\}$, где n обычно равно 2^{64}).
- Для множества шинглов A и случайной перестановки π можно определить min-hash функцию:

$$h_{\pi}(A) = \min_{i \in A} \pi(i)$$

- Так как π выбрана случайно, каждый элемент a множества A имеет одинаковую вероятность быть минимальным в отображении:

$$P(h_{\pi}(A) = \pi(a)) = \frac{1}{|A|}$$

- Пусть $\alpha = h_\pi(A \cup B)$, тогда $h_\pi(A) = h_\pi(B)$ тогда и только тогда, когда $\pi^{-1}(\alpha) \in A \cap B$:

$$P(h_\pi(A) = h_\pi(B)) = P(\pi^{-1}(\alpha) \in A \cap B) = \frac{|A \cap B|}{|A \cup B|}$$

- Тогда для любых множеств шинглов A и B :

$$P(h_\pi(A) = h_\pi(B)) = J(A, B)$$

Шинглы, Min-Hash

- Понятно, что одной перестановки не достаточно для обеспечения хорошего качества. На практике обычно используют 100-300 заранее подготовленных случайных перестановок.
- В таком случае мера схожести - доля совпавших хэшей.
- Как сравнить каждый документ с каждым? Подсчитать шинглы от min-hash шинглов (супершинглы). При совпадении одного из супершинглов необходимо проводить более точное сравнение.

- Один из вариантов LSH. Вероятностный метод понижения размерности.
- Малые изменения в исходном пространстве ведут к малым изменениям значения хэш функции.
- Обычно SimHash - n -битный вектор. И под малыми изменениями хэш функции понимаются малые отклонения в расстоянии Хэмминга.
- При обработке текстов, обычно $n = 64, 128, 256, 1024$.

SimHash. Random Hyperplane

- Один из вариантов построения SimHash функции.
- Необходимо сгенерировать n случайных плоскостей в исходном пространстве.
- В качестве значения соответствующего бита необходимо взять знак скалярного произведения вектора и нормали случайной плоскости.
- Расстояние Хэмминга в результирующем пространстве естественным образом связано с косинусной мерой в исходном.

SimHash. Простое построение

- Зафиксировать n , для которого имеется n -битная хэш-функция от термов(N -грамм) документа.
- Размерность исходного пространства будет равна n . Каждый терм задаётся в нём значением хэш-функции, в котором каждый бит проинтерпретирован как соответствующая координата (необходимо произвести сдвиг и растяжение, чтобы координаты принимали значения $\{-1, 1\}$).
- Для получения вектора документа в исходном пространстве, необходимо сложить векторы всех термов.
- Положить в качестве нормалей плоскостей единичные орты.
- В такой постановке очередной бит SimHash - знак соответствующей координаты.

SimHash. Поиск в большой коллекции

- Допустим, необходимо произвести поиск по коллекции SimHash на расстоянии, не большем $k - 1$.
- Разобьём каждый хэш на k частей. Тогда при поиске по крайней мере в одной из частей будет полное совпадение, для любого хэша из искомого множества.
- На этом основан самый популярный метод: разбить каждый хэш на k частей. По каждой из частей организовать отдельный индекс. Во время поиска спросить каждый индекс, результаты объединить и каждое значение проверить «в лоб».