Урок №5

Словари и нечёткий поиск

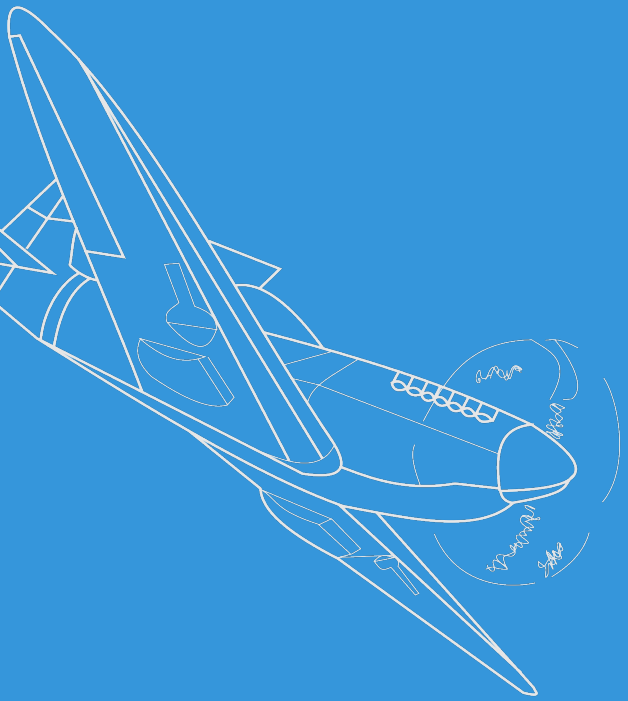
(основано на слайдах Андрея Калинина, Hinrich Schütze, Christina Lioma)

14 октября 2021 года

Антон Кухтичев

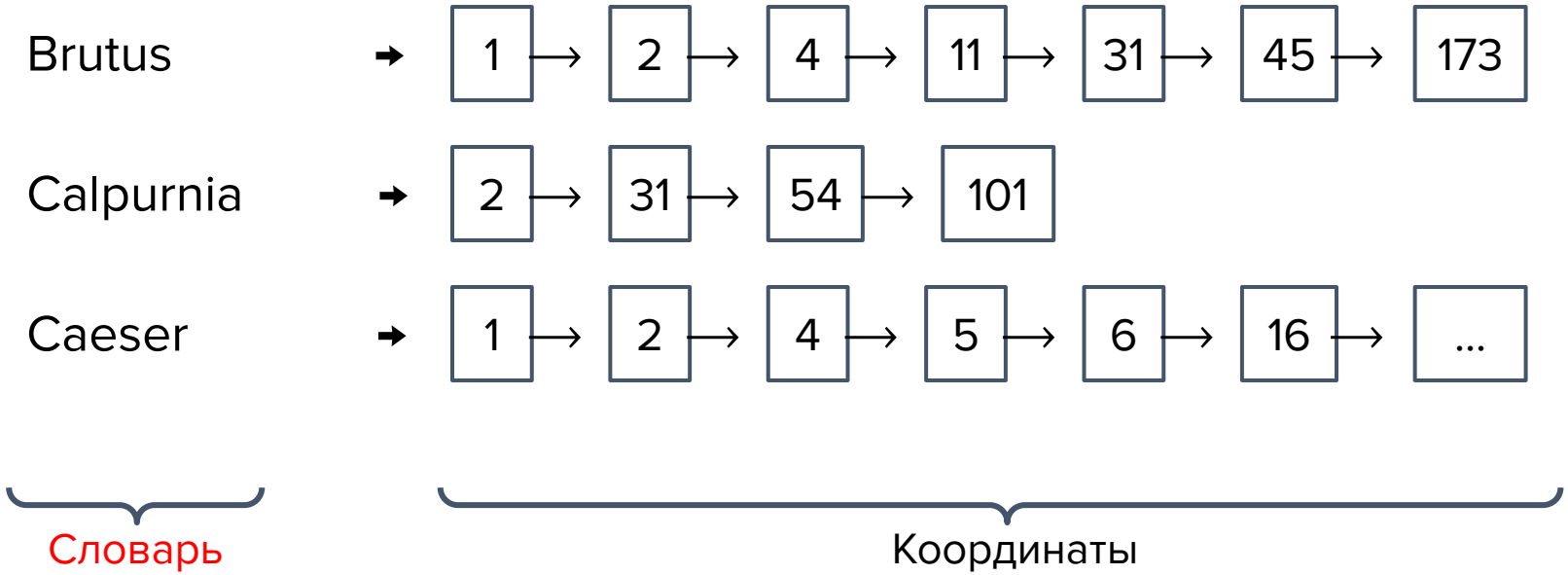
Содержание занятия

1. Словарь
2. Запросы с мета-символами
3. Проверка правописания
4. Soundex
5. Исправление запросов



Словарь

Обратный индекс



Словарь как массив

- Для каждого термина нужно сохранить:
  - количество документов (частотность)
  - указатель на координаты
  - ...
- На время допустим, что можно представить эту информацию в виде структуры фиксированной длины.
- Тогда можно использовать массив для хранения словаря.

Словарь как массив

	Термин	Частотность	Координатный блок
	a	656256	→
	aachen	65	→
	...	...	...
	zulu	221	→
объём:	20 байт	4 байта	4 байта

Как искать термин запроса q_i в этом массиве? То есть: какую структуру данных можно использовать, чтобы найти строку, в котрой находится q_i?

Структуры данных поиска терминов

- Два основных класса: хеши и деревья.
- Некоторые ИСП используют хеши, некоторые — деревья.
- Основные вопросы выбора:
  - Количество терминов фиксировано, или растёт?
  - Какие относительные частоты доступа к разным ключам?
  - Сколько разных ключей имеется?

Хеши

- Каждый термин хешируется в целое число.
- Боремся с коллизиями.
- Во время запроса: хешируем термин запроса, разрешаем коллизии, находим нужную строку в массиве.
- Плюсы:
  - Поиск в хеш-таблице быстрее, чем поиск в дереве.
  - Время поиска — константа.

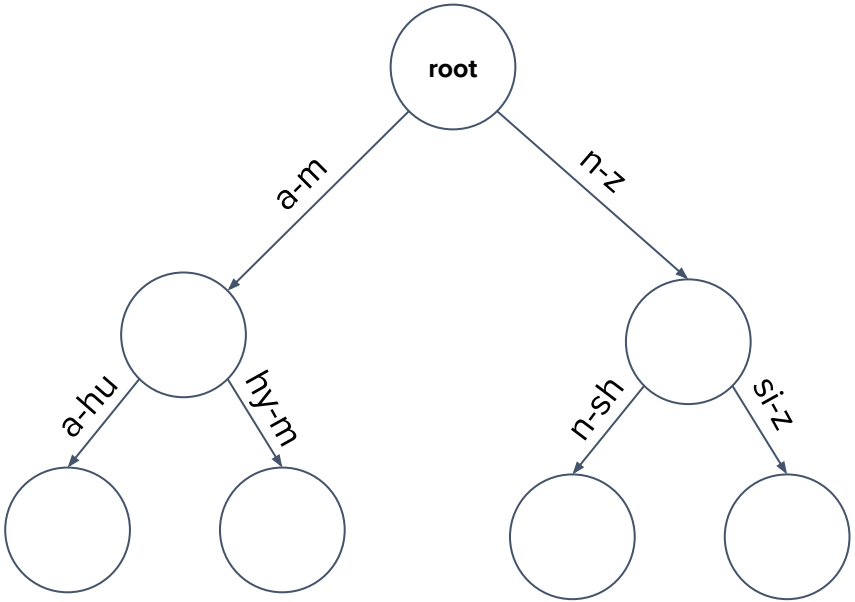
Хеши

- Минусы
  - Нельзя найти небольшие различия (resume и résumé)
  - Нельзя искать по префиксу (все термины, начинающиеся с
  - automat)
  - Для растущего словаря придётся время от времени всё рехешировать.
- Теоретически, можно сделать «морфологическую» хеш-функцию.

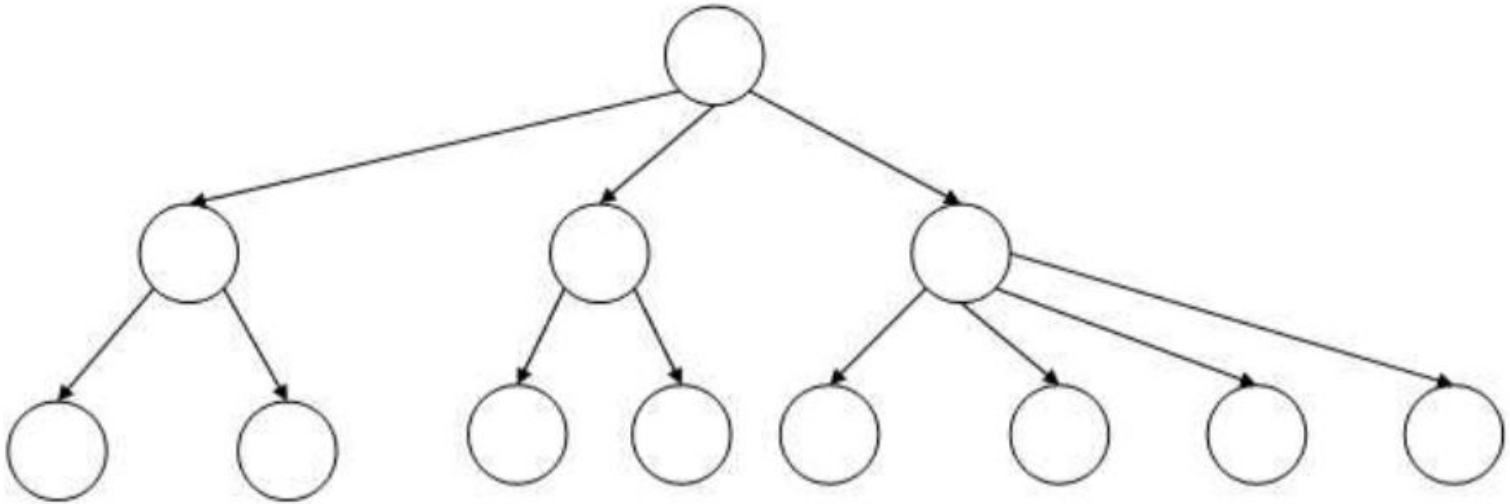
Деревья

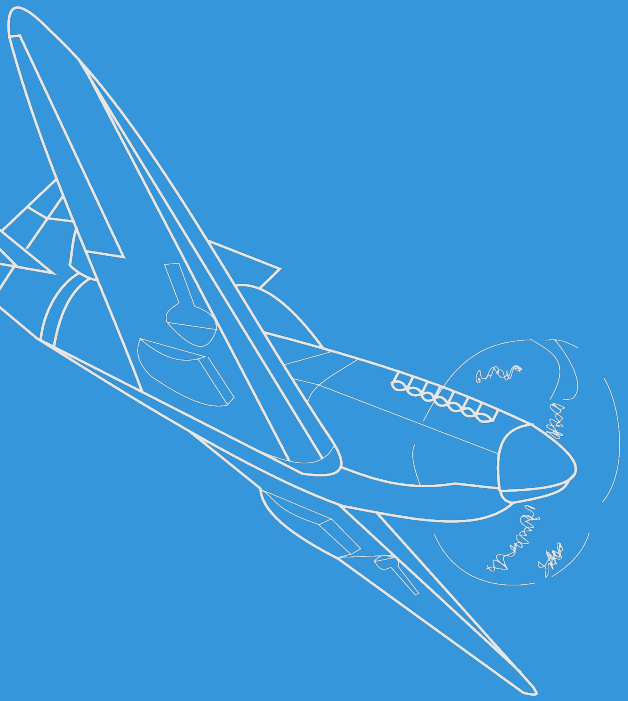
- Деревья позволяют искать термины с общим префиксом.
- Простейшее дерево — бинарное.
- Поиск медленее хешей, *O(log M)*, где M — размер словаря.
- *O(log M)* соблюдается для сбалансированных деревьев.
- Так же можно использовать B-деревья.

Бинарное дерево



B-дерево





Запросы с мета-символами

Запросы с мета-символами

- mon*: найти все документы, содержащие термин, начинающийся с mon
- Просто для B-дерева: найти все термины t, находящиеся в диапазоне mon ≤ t < moo
- *mon: найти все термины, заканчивающиеся на mon
  - Создаём дополнительное дерево, для терминов, записанных задом наперёд.
  - Теперь по этому дереву получаем термины t в диапазоне nom ≤ t < non
- Результат: множество терминов, подходящих под маску.
- Теперь нужно найти документы, содержащие любой из этих терминов.

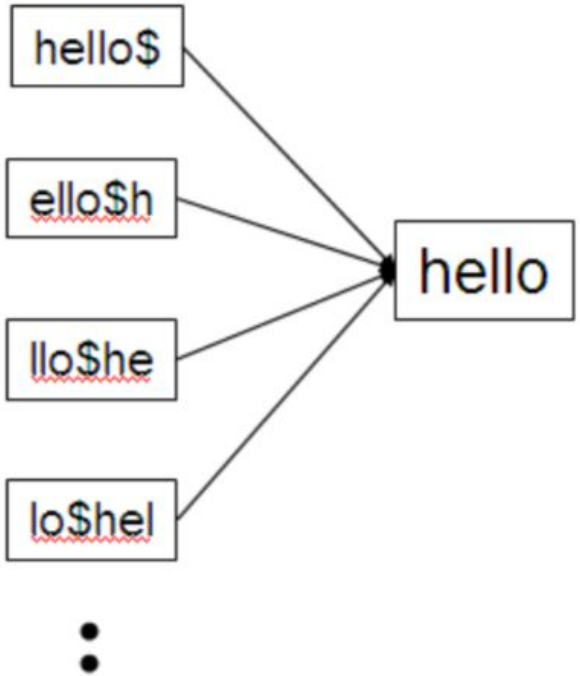
Как обработать * внутри термина

- Например: m*nchen
- Можно поискать m* и *nchen в B-деревьях и пересечь два полученных множества.
- Довольно расточительно.
- Альтернатива: индекс перестановок
- Основная идея: «переворачивать» каждый запрос с маской таким образом, чтобы * оказалась в конце.
- Хранить каждый переворот каждого термина в словаре, в том же B-дереве.

Индекс перестановок

- Для термина hello: добавим *hello$, ello$h, llohe, lohel, и o$hell* в B-дерево, где $ — специальный символ.

Отображение перестановок в термины



Индекс перестановок

- Итак, для `hello` храним: `hello$`, `ello$h`, `llo$he`, `lo$hel` и `o$hell`
- Тогда запросы
  - X, ищем X$
  - X*, ищем X*$
  - *X, ищем X$*
  - *X*, ищем X*
  - X*Y, ищем Y$X*
  - Например: для `hel*o` ищем `o$hel*`
  - Как обработать запрос X*Y*Z?

Поиск в индексе перестановок

- Прокрутить запрос так, чтобы * была справа.
- Искать как обычно.
- Однако: такой индекс как минимум учетверяет размер словаря (для английского языка, для русского — увеличит в 7-8 раз).

Индексы *k*-грамм

- Занимает меньше места, чем индекс перестановок.
- Индексируем все символьные *k*-граммы (последовательности из k символов) термина.
- 2-граммы часто называют биграммами.
- Например: из `April is the cruelest month` получим следующие биграммы: `$a ap pr ri il l$ $i is s$ $t th he e$ $c cr ru ue el le es st t$ $m mo on nt h$`
- `$` — специальный символ, обозначающий границу слова.
- Добавляем в новый индекс не термины, а биграммы.

3-граммный обратный индекс

k-граммные индексы

- Теперь у нас два разных вида обратных индексов.
- Есть индекс терминов-документов.
- И есть индекс *k*-грамм, чтобы находить термины по запросам, состоящие из *k*-грамм.

Выполнение запроса с метасимволами для биграмм

- Запрос `mon*` можно обработать так:

  `$m and mo and on`

- Так получим все термины с префиксом mon...
- ... но и много «ложных срабатываний», таких как `moon`.
- Их нужно отфильтровать, напрямую сравнивая термины с запросом.
- Оставшиеся термины нужно искать в индексе терминов-документов.
- *k*-граммный индекс и индекс перестановок
  - *k*-граммный индекс занимает меньше места.
  - Индекс перестановок не требует пост-фильтрации.

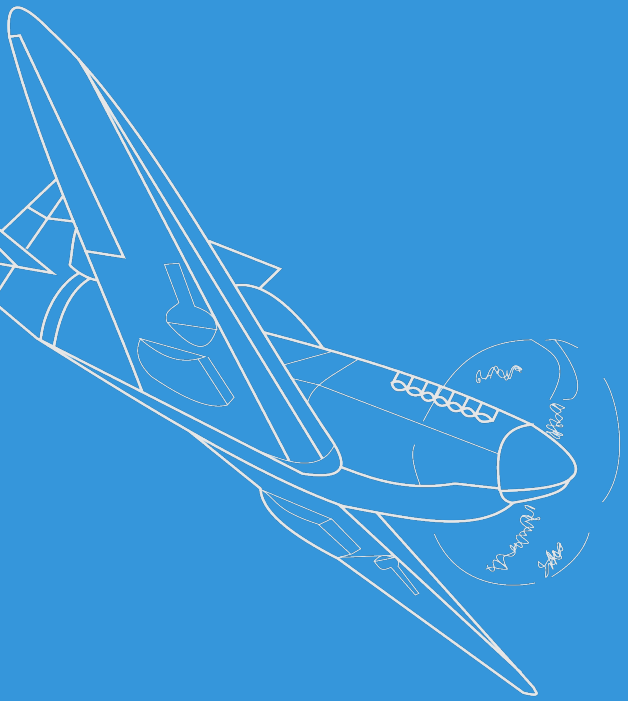
Упражнение

Почему у больших веб-поисков нет поддержки запросов с масками?

Упражнение

Почему у больших веб-поисков нет поддержки запросов с масками?

- Много слов.
- Увеличивается количество обрабатываемых терминов.
- Люди будут вводить меньше символов в словах.



Проверка правописания

Проверка правописания

- Два возможных применения:
  - Исправление документов.
  - Исправление запросов.
- Два разных метода:
  - Исправление отдельных слов
    - Проверяет каждое слово.
    - Не сможет исправить опечатки в словарных терминах, например an asteroid that fell form the sky
  - Контекстно-зависимое исправление
    - Обращает внимание на контекст, окружающие слова.
    - Может заменить ошибку в предыдущем примере (form/from)

Исправление документов

- Интерактивная коррекция документов не нужна.
- Используется в основном для распознанных документов (системы OCR).
- Обычно документы никак не изменяются.

Исправление запросов

- Самое простое: исправление отдельных слов
- Допущение 1: имеется список «правильных слов».
- Допущение 2: есть способ вычисления расстояния между словом с опечаткой и правильным словом.
- Тогда простейший алгоритм возвращает «правильное» слово с наименьшим расстоянием к слову с опечаткой.
- Например: informaton ➔ information
- В качестве списка правильных слов можно использовать словарь ИПС.
- Почему это плохо?

Альтернативные источники «правильных» слов

- Стандартные словари (Зализняк)
- Технические словари (для специализированных ИПС)
- Отфильтрованные словари корпуса ИПС

Расстояния между словами

- Две альтернативы:
  - Расстояние Левенштейна
    - Взвешенное расстояние Левенштейна
  - Пересечение k-грамм

Расстояние Левенштейна

- Расстояние между строками s_1 и s_2 — количество элементарных операций редактирования, нужных для преобразования s_1 в s_2.
- Растояние Левенштейна: операции вставки, удаления и замены.
  - dog-do: 1
  - cat-cart: 1
  - cat-cut: 1
  - cat-act: 2
- Расстояние Левенштейна-Дамерау: добавлена операция перестановки двух рядом стоящих символов.
  - cat-act: 1

Вычисление расстояния Левенштейна

		f	a	s	t
	0	1	2	3	4
c	1	1	2	3	4
a	2	2	1	2	3
t	3	3	2	2	2
s	4	4	3	2	3

Расстояние Левенштейна: вычисление

Algorithm Edit distance

Input: $\alpha = \alpha_1 \dots \alpha_n$ and $\beta = \beta_1 \dots \beta_m$

```
1:  for i ← 0 to n  do
2:      D_{i,0} ← i;
3:  end for
4:  for j ← 0 to m  do
5:      D_{0,j} ← j;
6:  end for
7:  for i ← 1 to n  do
8:      for j ← 1 to m do
9:          t ← (α_i = β_j)? 0 : 1;
10:         D_{i,j} ← min{D_{i-1,j-1} + t, D_{i,j-1} + 1, D_{i-1,j} + 1};
11:     end for
12:  end for
13. return D_{n,m}
```

Взвешенное расстояние

- Аналогично предыдущему, но веса операций зависят от символов.
- Нужно для учёта клавиатурных опечаток, например m более вероятно ошибочно напечатать как n, чем как q.
- Поэтому, замена m на n — меньшее расстояние, чем замена на q.
- Теперь нужна матрица весов.
- Так же нужно добавить в алгоритм учёт этих весов.

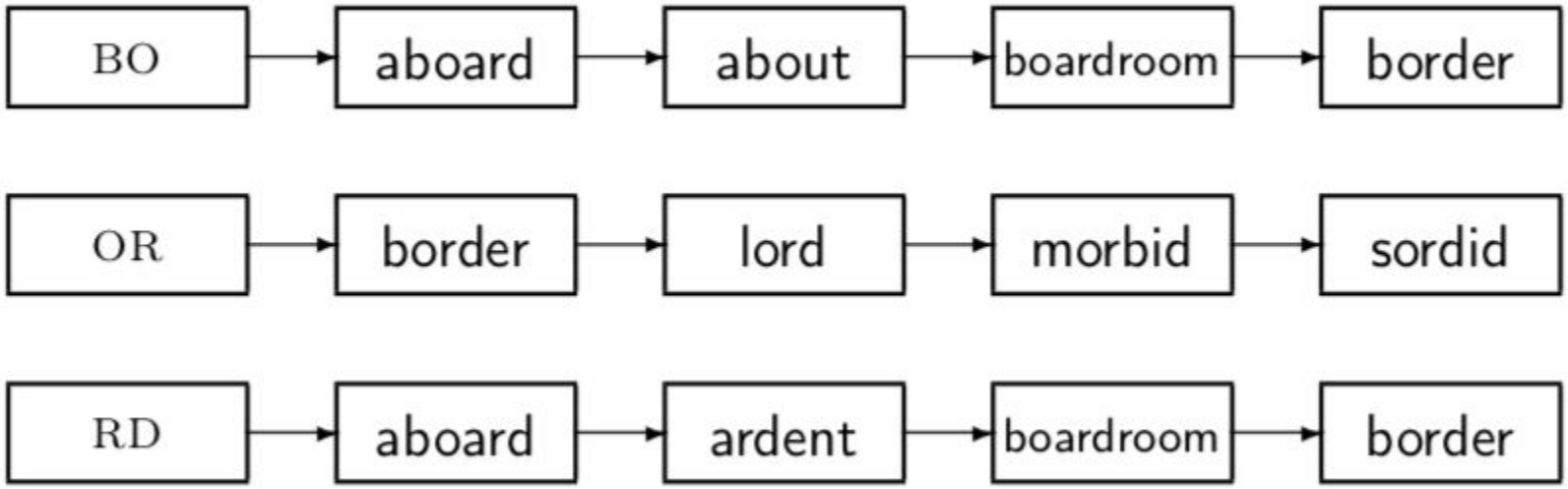
Исправление опечаток с учётом весов

- Для данного запроса перебрать все строки на заданном расстоянии.
- Пересечь это множество со списком «правильных» слов.
- Предложить термины из пересечения пользователю.
- Или автоматически исправить запрос.
- Что лучше?

k-граммный индекс для исправления опечаток

- Перебрать все k-граммы из термина запроса
- Например: биграммный индекс, слово с опечаткой `bordroom`
- Биграммы: `bo, or, rd, dr, ro, oo, om`
- Используем индекс k-грамм для получения «правильных» слов
- Устанавливаем предел по количеству совпавших k-грамм
- Например, нужны только такие термины, которые отличаются не больше чем по трём k-граммам.

Исправление опечатки для bordroom



Пример с триграммами

- Проблема: фиксированное количество отличающихся k-грамм по разному работает для слов разной длины.
- Например, правильное слово `november`
  - Триграммы: `nov`, `ove`, `vem`, `emb`, `mbe`, `ber`
- И запрос `december`
  - Триграммы: `dec`, `ece`, `cem`, `emb`, `mbe`, `ber`
- Таким образом, 3 триграммы пересекаются (из 6 у каждого термина)
- Нужна нормализованная метрика.

Коэффициент Жаккара

- Метрика пересечения двух множеств.
- Два множества, A и B
- Коэффициент Жаккара: $\frac{|A \cap B|}{|A \cup B|}$
- A и B не обязаны иметь одинаковый размер.
- Результат — число между 0 и 1.
- december/november — какой коэффициент Жаккара?
- В проверке правописания можно использовать в качестве лимита, исправлять только для значений коэффициента > 0.8.

Контекстно-зависимая проверка (1)

- Пример: `an asteroid that fell form the sky`
- Как можно исправить слово form?
- Одна из идей: статистика вхождений.
  - Получить «правильные» термины, близкие к каждому термину.
  - Для запроса `flew form munich`: flea для flew, from для form, munch для munich

Контекстно-зависимая проверка (2)

- Поискать все возможные варианты с одним исправленным словом:
  - Сначала “`flea form munich`”
  - Затем “`flew from munich`”
  - Потом “`flew form munch`”
- Правильный запрос “`flew from munich`” вернёт больше всего результатов.

- Допустим, у нас есть 7 вариантов для flew, 20 для form и 3 for munich, сколько запросов на проверку получится?

Контекстно-зависимая проверка (3)

- Алгоритм на предыдущем слайде не очень эффективен.
- Лучшая альтернатива — исследовать запросы, а не документы.

Общие проблемы исправления опечаток

- Пользовательский интерфейс:
  - Заменять автоматически или предлагать?
  - Возможно, Вы имели в виду подходит только для одного предложения.
  - Что делать с большим количеством вариантов?
  - Компромисс между простым или гибким интерфейсом.
- Затраты:
  - Потенциально очень затратно.
  - Обрабатывать выборочно?
  - Например, только для запросов, вернувших мало результатов.
  - Исправление опечаток для крупных ИПС достаточно быстро работает, чтобы обслуживать каждый запрос.

Рекомендуемая литература

[Введение в информационный поиск | Маннинг Кристофер Д., Шютце Хайнрих]

Для саморазвития (опционально)

[Чтобы не набирать двумя пальчиками]

Спасибо за внимание!

Антон Кухтичев

a.kukhtichev@mail.ru

@toshunster