Домашнее задание №2

Информационный поиск. 6 курс. Осенний семестр.

Горбунов Егор Алексеевич

11 ноября 2016 г.

Задание №1 Рассмотренные методы исправления опечаток не работают напрямую при пропуске пробела (например, informationretrieval). Опишите, как исправлять такие опечатки (не обязательно на основе рассмотренных методов).

Решение:

(a) Если допустить, что пользователь редко по-случайности склеивает более 2-ух слов, то можно хранить, помимо словаря слов, словарь коллокаций длины 2. Таким образом теперь для исправления пропуска ***одного*** пробела можно использовать обычное редакционное расстояния, которое должно теперь учитывать, что пробел — это тоже символ. Минус — сильно возрастает размер словаря и метод становится непрактичен (если считать, что в Английском языке около 400000 слов, что умещается в несколько мегабайт, то все коллокации займут около пары терабайт).

(b) Будем сканировать входное слово w из запроса q слева направо до тех пор, пока полученный префикс p не будет словом из словаря. Далее отрезаем этот префикс от слова и повторяем те же действия с оставшейся строкой и таким образом получаем места, где нужно расставлять пробелы. У такого метода есть проблемы:

- Не учитывает опечаток помимо пробелов
- Что делать, если дошли до конца w, а набранный префикс не в словаре? Нужно добавлять откат назад и пытаться расширить уже существующее слово из словаря, равное какому-то префиксу.
- Имеет тенденцию разбивать слово на более короткие куски, т.е., если в словаре есть слова ***info*** и ***rmation***, то ***informationretieval*** превратится в ***info rmation retrieval***. В данном случае лучше предпочитать большие части, т.к. они, скорее всего, несут больше смысла.

(c) Теперь придумаем что-нибудь более содержательное. Имея слово $w[1, n]$ (длины n) мы хотим найти самое ближайшее к нему исправление с возможными разбиениями (вставкой пробела). Рассмотрим такое оптимальное разбиение $w[1, n]$: $(w[1, i], w[i + 1, k], w[k + 1, n])$ мы точно можем заключить, что оптимальное разбиение, например $w[i + 1, n]$ — это $(w[i + 1, k], w[k + 1, n])$. Эту

идею можно формализовать в виде задачи динамического программирования:

$$d(i,j) \text{ — стоимость редактирования куска } w[i,j]$$
$$d(i,i) = \infty \text{ (запрет разбиения на отдельные буквы)}$$
$$d(i,j) = \min\left\{\min_{i\le k<k}\left(d(i,k)+d(k+1,j)+1\right), \mathrm{editDist}(w[i,j])\right\}$$

Тут $\mathrm{editDist}$ — это обычное редакционное расстояние, которое находится по словарю (т.е. поиск самого близкого слова к данному). Ответ на задачу считается последовательно по возрастанию размера куска и будет записан в $d(1,|w|)$. Таким образом мы и находим оптимальное разбиение пробелами. Единичка в выражении $d(i,k)+d(k+1,j)+1$ — это штраф за разбиение. Заметим, что можно добавить запреты на очень короткие слова путём задания больших значений для $d(i,i+len)$. По сложности данный метод получается не сильно хуже, чем обычное редакционное расстояние (куб от длины слова (она не очень большая) на время вычисления editDistance). Вместо редакционного расстояния можно использовать и коэффициенты Жаккара.

Задание №2 Мы рассмотрели два типа методов для рекомендации запросов, аналогичных заданному запросу:

a) Рекомендовать запросы, встречающиеся в одной сессии с заданным запросом.

b) Рекомендовать те запросы, у которых множество кликнутых результатов сильно пересекается с аналогичным множеством для заданного запроса.

Какие еще методы для рекомендации запросов вы можете предложить?

Решение: Приведённые методы рекомендации не привлекают глубокого анализа пользовательского запроса, а опираются не действие пользователей. Поэтому хочется предложить методы, которые учитывают контекст запроса. При помощи методов NLP можно попытаться извлечь из запроса сущности, такие как фильмы, музыку и места. Далее на помощь нам приходят графе сущностей и отношений между ними: по фильмам мы можем находить актёров и прочих личностей, по музыке мы можем находить композиторов и другие их произведения, по местам мы можем находить похожие по типу и по расположению на карте места. Все эти данные могут служить рекомендациями.

Задание №3 Вы планируете использовать следующие методы поиска: модель векторного пространства с весами TF-IDF, BM25, языковую модель. Какую минимальную информацию должен содержать индекс, чтобы поддерживать эффективное использование этих методов? Какую информацию нет смысла хранить в индексе? Как ее нужно хранить? Дайте развернутый ответ.

Решение:

TF-IDF Минимальная информация для эффективного использования:

- Векторные представления каждого документа (считать векторные представления на лету было бы очень затратно)

- Значения TF-IDF для всех возможных слов (термов), чтобы считать векторное представление запроса быстро
- Обратный индекс из термов в документы, ясное дело, чтобы быстро формировать множества документов, в которых будет присутствовать слово из запроса. Перебирать все документы не нужно, т.к. косинусное расстояние у документов не содержащих ни одного слова из запроса с запросом будет заведомо равным 0

Не нужно хранить, например, векторные представления для всевозможных запросов. Хранить просто: всем документам раздать id и устроить мапу из id в векторное представление. В обратном индексе хранить map из терма в список id. И так же хранить мап из слов в tfifd для них. Сами документы, конечно, нужно уметь доставать по id.

BM25 Минимальная информация для эффективного использования (и её хранение):

- Храним $\mathrm{tf}(t, d)$ для каждого терма и документа. Храним в двухуровневой мапе (на первом уровне может быть массив): $\mathrm{tf}[d][t] := \mathrm{tf}(t, d)$ Разумеется тут используем разреженные структуры данных.
- Храним значения $k_1[(1-b) + b\frac{\mathrm{dl}(d)}{\mathrm{dl}_{ave}}]$ для каждого документа в массиве $\mathrm{arr}[\mathrm{doc}_{id}]$, чтобы не выполнять арифметические операции каждый раз
- Храним коэффициент $k = k_1 + 1$ (это и пункт выше позволяют упростить множитель в формуле BM25)
- Для терма храним $\mathrm{df}(t)$ в мапе.
- Обратный индекс как в прошлом пункте

Не нужно хранить векторное представление документа.

LM Что и как храним:

- В разреженных структурах храним вероятности $P(t|M_d)$ для каждого документа.
- Обратный индекс аналогично пунктам выше

Сами языковые модели мы не храним, т.к. они используются исключительно для получения вероятностей $P(t|M_d)$

Задание №4 Отранжируйте документы из таблицы 1 по запросу “car insurance” с использованием модели векторного пространства и весов TF-IDF.

Corpo	idf	tf		
		Документ 1	Документ 2	Документ 3
car	1.65	27	4	24
auto	2.08	3	33	0
insurance	1.62	0	33	29
best	1.60	14	0	17

Таблица 1: Частота слов в документах и обратная документная частота слов.

Решение: Вектора: (для запроса tf(car) = tf(insurance) = 1). Каждый элемент вектора получен умножением tf(word, doc) * idf(word)

	car	auto	insurance	best	норма вектора
Документ 1	44.55	6.24	0	22.4	50.25
Документ 2	6.6	66.64	53.46	0	85.68
Документ 3	39.6	0	46.98	27.2	67.19
Запрос	1.65	0	1.62	0	2.312

Scores (косинусы):

- $score(d1, q) = \frac{73.5}{50.25*2.312} = 0.633$
- $score(d2, q) = \frac{97.5}{85.68*2.312} = 0.492$
- $score(d3, q) = \frac{141.4}{67.19*2.312} = 0.91$

Ранжирование: D3, D1, D2

Задание №5 Рассмотрим следующий запрос и три результата.

Q information retrieval course

D1 Information Retrieval and Web Search

D2 Introduction to Information Retrieval

D3 Text Retrieval and Search Engines

Результаты 1 и 3 - это страницы соответствующих курсов, поэтому пользователь пометил их как релевантные. Результат 2 - это страница с книгой, поэтому пользователь пометил его как нерелевантный. Примените алгоритм Роккио и выпишите вектор запроса после учета обратной связи по релевантности. Элементы вектора перечислите в алфавитном порядке. Считайте, что компоненты векторов содержат только частоты слов (без обратной документной частоты и нормировки). Параметры алгоритма Роккио: $\alpha = 1, \beta = 0.75, \gamma = 0.25$.

Решение: TODO

Задание №6 Выпишите формулу BM25 для длинных запросов. Опишите ее составляющие. Каким образом каждая составляющая влияет на ранжирование (т.е. что происходит с ранжированием результатов при изменении каждой из составляющих)?

Решение: TODO

Задание №7 Пусть бинарная случайная величина X_t - это индикатор того, что слово t встречается в документе (т.е. $X_t = 1$, если слово t есть в документе, и $X_t = 0$, если слова t нет в документе). $P_t = P(X_t = 1 \mid d)$ - это вероятность того, что слово t встречается в документе d. Примените метод максимального правдоподобия (MLE) для формального вычисления P_t и покажите, что $P_t = \frac{tf(t,d)}{dl(d)}$, где $tf(t, d)$ - это частота слова t в документе d, а $dl(d)$ - это длина документа d.

Решение: TODO

Задание №8 Рассмотрим коллекцию из двух документов.

D1 A language model is a probability distribution over words or sequences of words.

D2 A language model is used in many natural language processing applications.

Выпишите сглаженную униграмную языковую модель для каждого документа. Используйте сглаживание Jelinek-Mercer с параметром $\lambda = 0.5$. Отранжируйте эти документы по запросу «many words».

Решение: Сглаженная модель:

$$P_s(t|M_d) = 0.5(\frac{tf(t,d)}{dl(d)} + \frac{cf(t)}{cl})$$

Разные величины:

- $dl(D1) = 13, dl(D2) = 11$
- $cl = 13 + 11 = 24$
- Словарь (cf в скобках): a(3), language(2), model(2), is(2), probability(1), distribution(1), over(1), words(2), or(1), sequences(1), of(1), used(1), in(1), many(1), natural(1), processing(1), applications(1)

	D1	D2
tf(a)	2	1
tf(language)	1	1
tf(model)	1	1
tf(is)	1	1
tf(probability)	1	0
tf(distribution)	1	0
tf(over)	1	0
tf(words)	2	0
tf(or)	1	0
tf(sequences)	1	0
tf(of)	1	0
tf(used)	0	1
tf(in)	0	1
tf(many)	0	1
tf(natural)	0	1
tf(processing)	0	1
tf(applications)	0	1

Модели документов:

	D1	D2
P(a)	0.139	0.108
P(language)	0.08	0.087
P(model)	0.08	0.087
P(is)	0.08	0.087
P(probability)	0.059	0.02
P(distribution)	0.059	0.02
P(over)	0.059	0.02
P(words)	0.118	0.041
P(or)	0.059	0.02
P(sequences)	0.059	0.02
P(of)	0.059	0.02
P(used)	0.02	0.066
P(in)	0.02	0.066
P(many)	0.02	0.066
P(natural)	0.02	0.066
P(processing)	0.02	0.066
P(applications)	0.02	0.066

Ранжирование: D1, D2 (это очевидно!)