Домашнее задание 6

В данном домашнем задании Вам предстоит реализовать автоматическое исправление опечаток в запросах пользователей.

1. Датасет

Для оценки качества алгоритма исправления опечаток, Вам предоставляется файл *queries.tsv.gz*. В каждой строке файла записаны два запроса – исходный и исправленный. Для простоты, оба запроса будут иметь одинаковое количество слов и отличаться незначительно. Зачастую исходный и исправленный запрос совпадают, что означает что исправлять такой запрос не требуется.

Для составления словаря и обучения языковых моделей Вам предоставляется небольшой корпус текста, неслучайная выборка из большой английской википедии в файле *train.bz2*. Этот файл содержит примерно 5 млн строк или 80 млн слов. Каждая строка – одно предложение без знаков препинания.

Использование других словарей и корпусов запрещено.

2. Поиск близких слов

Требуется научится быстро находить список из сотни слов, которые незначительно отличаются от заданного слова.

Не стоит перебирать все слова словаря – займёт слишком много времени.

Для ускорения перебора предлагается создать триграммный индекс – для каждой буквенной триграммы храним список слов, в которых она есть. Тогда для поиска похожих на данное слово найдем слова большим количеством совпадающих триграмм.

Совет 1: стоит сделать отельный индекс для каждой длинны слова и использовать только те индексы, в которых лежат слова близкие по длине к исходному.

Совет 2: для выделения триграмм стоит обрамить слово спецсимволом, чтобы триграммы на концах слова отличались от оных в середине.

Любые другие алгоритмы, улучшающие качество за разумное время (хождение по бору с ошибками, перебор ошибок) – не возбраняются.

Чтобы оценить качество полученного алгоритма, используйте запросы из *queries.tsv.gz*. Отберите только отличающиеся слова в исправленном и исходном запросах. Проверьте, что для слова в исходном запросе, исправленное слово будет в списке ближайших выданном вашим алгоритмом. Если это выполняется для всех или почти всех пар – успех.

Не побрезгуйте кешировать результат работы этого алгоритма, чтобы дальнейшая работа протекала быстрее.

3. Языковая модель

Языковая модель – модель, которая по тексту оценивает вероятность того, что он мог появиться в языке.

Постройте простую n-грамную языковую модель с использованием корпуса текстов *train.bz2*. Для этого рассчитайте количество вхождений каждой n-граммы в корпус текста. Если взять n=2, то размера оперативной памяти вашего компьютера должно будет хватить.

Воспользуйтесь каким-нибудь методом сглаживания, чтобы не получать нулевую вероятность для неизвестных n-грамм. Также, чтобы вероятности слов, которых нет в словаре, были отличны от нуля, можно примешать побуквенную m-граммную модель.

Совет N: если количество оперативной памяти прижмёт, можно хранить строки в виде байт – один раскодированный символ занимает больше памяти чем один байт, при этом для английского текста почти всегда один символ кодируется одним байтом.

Чтобы оценить качество полученной модели, используйте запросы из *queries.tsv.gz*. Сравните вероятность, которую выдает ваша модель для исходных и исправленных запросов. Хорошая модель выдаёт исправленному запросу большую вероятность.

Советую сохранить полученную модель на диск – а случае чего, чтение статистик с диска, может быть быстрее расчёта оных с нуля.

4. Модель ошибок

Модель ошибок – модель которая по исходному и исправленному запросу оценивает вероятность того, что такая ошибка могла быть допущена.

Рассчитайте простую модель ошибок на основе расстояния Дамерау-Левенштейна, то есть модифицированного Левенштейна, который считает перестановку соседних букв за одну ошибку.

5. Олтугеза

Объедините результат работы предыдущих пунктов в единый алгоритм исправления опечатки для запроса.

Примерный план:

1. Для слов запроса генерируем список ближайших слов-кандидатов (для всех, даже словарных слов).
2. Собираем список кандидатов-запросов (эвристически, чтобы не сделать экспоненциальное время выполнения)
3. Для каждого кандидата считаем итоговый объединенный score на основе языковой модели и модели ошибок для данного кандидата (не обязательно сумма или произведение, можно объединение любой сложности).
4. Выдаём гипотезу с наибольшим score.
5. ???
6. Profit

Итоговое качество меряем на примерах из *queries.tsv.gz*.

Для отладки проблем с качеством имеет смысл научится понимать на каком этапе теряется правильная гипотеза для каждого примера. Например, если правильное исправление есть в списке кандидатов (п. 2), но не выбирается как лучшая – стоит крутить языковую модель, модель ошибок и их объединение.