Шпаргалка
Домашняя работа по теме: «Поиск дубликатов»

Постановка задачи

Дано: коллекция документов
Задание: найти все пары дубликатов, в каждой из которых документы похожи более, чем на определенную величину (75%)

Проверка решения

Проверка: проверка задания состоит из 3 этапов. На каждом этапе — новая коллекция документов (коллекция с первого этапа была выдана для тренировок, её успешно подтверждённая обработка даст вам первые 2 балла). Результаты работы задачи (пары дубликатов) нормируются и сравниваются с опорным вариантом, для двух этих множеств считается мера Жаккара. Если JC >= 0.9 , то этап считается успешно пройденным
Входные данные: несколько файлов формата .gz, внутри каждого — набор протобуфов (google-protobuf), по одному протобуфу на каждый документ. Протобуфы содержат поля: .url (его надо выводить для пары дубликатов), .body («сырой» html), .text (уже извлеченный текст, если есть, что извлекать).

Требования к решению

От студента требуется прислать архив, который должен быть в любом формате из следующего списка: .tar.gz , .tgz , .tar , .tar.bz2 .
В архиве должен быть как минимум 1 файл на корневом уровне — run.sh . Внутри этого скрипта должен быть реализован вызов кода, который и занимается поиском дубликатов. В случае, если требуется доставить какие-то модули, то там же (на корневом уровне архива) должен быть исполняемый файл preinstall.sh с соответствующими инструкциями (root и sudo без пароля уже есть). Однако его наличие совсем не обязательно.

Вызов кода:
$\$./run.sh \quad ./input\_file_1.gz \quad ./input\_file_2.gz \quad ...$, где $input\_file_i.gz$ — тестовые архивы с документами.
Результат работы должен писаться в стандартный output в формате: $< url_1 > \quad < url_2 > \quad < value >$, где url_i — урл документа (присутствует в исходном протобуфе), $value$ — степень подобия этих документов (больше 0.75). Порядок документов внутри пары (так же как и порядок пар) — не важен. Логи и прочие лишние данные в выводе будут считаться неправильно найденными парами дубликатов.

Технические моменты

Автоматическая проверка

To: duplicates_itmo@mail.ru
Subj: [HW3] BD-22, Иванов Иван
Архив должен быть приложен к письму. Текст не обязателен.
В ответ придет письмо, сообщающее о том, смог ли запуститься тест и, если да, то с каким результатом.

Окружение

Ubuntu 14.04 (x86_64)
Python 2.7
modules: numpy (1.11.2), google protobuf (3.1.0), mmh3, html2text

Ограничения

Время выполнения каждого из 3 этапов: по 20 минут
Ограничение по памяти: 4Гб
Все задания проверяются последовательно. Если ваше задание долго проверяется, то этому может быть 2 причины: технические проблемы у демона или слишком большая очередь. В последнем случае можете говорить «спасибо» своим коллегам :)

Оценки

По результатам всех попыток в качестве оценки выставляется максимальный из всех полученных баллов. Поэтому стараться прислать *по-*

следним самое правильное решение не надо — достаточно его просто хотя бы раз прислать.

Разбалловка

1 этап (открытый)	→ +2 балла
2 этап	→ +4 балла
3 этап	→ +4 балла
Итог:	10 баллов

Дедлайн

12 марта (включительно)

После этого числа все баллы превратятся в тыкву.

Дедлайн определяется по дате письма. Т.е. если вы пришлете письмо в срок, а демон не успеет его обработать — не волнуйтесь, всё зачтётся :)

Советы по решению

- текст нормализовывать не обязательно
- чтобы влезть по памяти (и в алгоритме Бродера, и вообще) и получить результат, максимально близкий к опорному варианту, советую сделать следующее:
  1. шинглы считать не побуквенно, а пословно
  2. размер 1 шингла — 5слов
  3. размер свёртки (minshingle) — 20 перестановок
  4. ... и вообще можно использовать ту заготовку, что я давала на лекции, — бо̀льшая часть задания там уже выполнена. В том числе вычисление шинглов и построение миншингла
  5. меру подобия для 2 документов считать надо через меру Жаккара, т.е. $\frac{(количество\_совпавших\_позиций)}{(количество\_совпавших) + 2 * (количество\_несовпавших)}$. Другими словами для 75% подобия необходимо совпадение не менее 18 шинглов

6. какую сигнатуру использовать?
   (a) выбор репрезентативного множества через корзины по модулю — этот подход реализован в заготовке, там допустимы сортировки внутри репрезентативного множества
   (b) построение сигнатуры свёртками — этот подход рассказывался на лекции, каждый элемент сигнатуры там привязан к позиции в ней

   для построения опорного варианта использовались множества с допустимыми сортировками