### Московский авиационный институт

**Факультет прикладной математики и физики**

**Лабораторная работа №4**

**по курсу:**

**«Информационный поиск»**

**по теме:**

**«Булев поиск»**

**2 семестр**

Студент: Ахмед С. Х.

Преподаватель: Калинин А. Л.

Группа: 8О-106М

**Москва, 2019 г**

Постановка задачи

Нужно реализовать ввод поисковых запросов и их выполнение над индексом, получение поисковой выдачи. Синтаксис поисковых запросов:

• Пробел или два амперсанда, «&&», соответствуют логической операции «И».

• Две вертикальных «палочки», «||» – логическая операция «ИЛИ»

• Восклицательный знак, «!» – логическая операция «НЕТ»

• Могут использоваться скобки. Парсер поисковых запросов должен быть устойчив к переменному числу пробелов, максимально толерантен к введённому поисковому запросу. Примеры запросов:

• [ московский авиационный институт ]

• [ (красный || желтый) автомобиль ]

• [ руки !ноги]

Так же должна быть реализована утилита командной строки, загружающая индекс и выполняющая поиск по нему для каждого запроса на отдельной строчке входного файла.

В отчёте должно быть отмечено:

• Скорость выполнения поисковых запросов.

• Примеры сложных поисковых запросов, вызывающих длительную работу.

• Каким образом тестировалась корректность поисковой выдачи

Ход решения

Итак, я решил воспользоваться встроенным типом данных в Python: множества, для которых определены операции пересечения и объединения. Также я решил воспользоваться деревом выражений в узлах которых определены операции AND и OR соответсвующие && и || соответсвенно. Но предварительно решил совершить препроцессинг данных, обработать случай отрицания целого выражения, замена пробелов && там где нужно и обработка особых случаев

Сам алгоритм поиска тривиален. Первое мы нормализуем слова, дальше зависит от представления словаря и блоков. В случае дерева вычисляю хэш строки, затем вытаскиваю значения по ключу и смотрю на операцию, если вижу операцию || объединяю блоки операндов, если && пересекаю, если вижу отрицание выполняю операцию вычитания из списка всех блоков.

Одной из причин реализации дерева выражения является наличия приоритета вычисления (наличие скобок), а также скорость прохода по дереву.

В ходе работы выяснилось, что скорость выполнения запроса оказалось связанным с длинной поискового запроса (как мне показалось странным, по сути я предполагал, что это будет зависеть от скорости выполнения операции пересечения (для сложных поисковых запросов)) Однако, операции пересечения и объединения являются встроенными в питон, то они реализованы крайне эффективно

Примеры поисковых запросов находятся в гит репозитории, вместе с замерами по времени.

Проверка результатов осуществлялась с помощь утилит командной строки grep и т.п для поиска вхождения слов. Также заранее был произведен отбор статей, как статьи для теста