Московский Авиационный Институт
(Научный Исследовательский Институт)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Отчёт по лабораторной №4 по курсу «Информационный поиск»

Выполнил: Ефименко Н. А.

Группа: 8О-106М

Преподаватель: Калинин А.Л.

Москва, 2020

ЛР4: Булев поиск

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.

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.

Этапы выполнения:

- Реализация основных поисковых операций над векторами чисел (объединение, пересечение, отрицание)
- Реализация механизма разбора входной строки и ее вычисления
- Полный цикл выполнения поискового запроса
- Тестирование
- Сравнение полученного решения с результатами полученными в ЛР2: Оценка качества поиска
- Анализ полученного решения, недостатки и методы их устранения

Реализация поисковых операций И, ИЛИ, НЕ.

НЕ

Конструирование структуры set из индексов документов которе не содержат токен. Операция является сложной, может колоссально увеличивать множество.

ИЛИ

Конструирование массива содержащего как элементы одного так и другого. Операция может увеличивать исходное множество.

И

Конструирование массива в который входят элементы, являющиеся общими для двух входных массивов. Наиболее оптимальная операция, исходное множество не увеличивает.

Вычисление

Входная строка имеет инфиксную запись. При помощи алгоритма сортировочной станции Дейкстры преобразуем запрос в постфиксную запись. Эта запись дает нам простой способ выполнения операций над операндами учитывая арность. приоритеты и ассоциативность операций

```
def __operatorsFactory(
  self,
  negation = lambda x: x,
  conjunction = lambda x: x,
  disjunction = lambda x: x) -> dict:
    return {
    '!': self.__Op(prec=4, assoc=self.__RIGHT, arity=1, calc=negation),
    '&': self.__Op(prec=3, assoc=self.__LEFT, arity=2, calc=conjunction),
    '|': self.__Op(prec=2, assoc=self.__LEFT, arity=2, calc=disjunction)}
```

```
def parsed_terms(self, query, adapter):
    res = []
    term = ''
    for s in query:
      if not (s in self.OPERATORS or s in "()"):
        term += s
      elif term:
        if len(term.strip()) > 0:
          res.append(adapter(term.strip()))
        term = ''
    if term:
      res.append(adapter(term.strip()))
    return res
```

```
def __calc(self, polish):
    stack = []
    for token in polish:
      if token in self.OPERATORS:
        operator = self.OPERATORS[token]
        if (operator.arity == 1):
          e = stack.pop()
          stack.append(self.OPERATORS[token].calc(e))
        else:
          x = stack.pop()
          y = stack.pop()
          stack.append(self.OPERATORS[token].calc(x, y))
      else:
        stack.append(token)
    return list(stack[0])

def build(self, adapter):
  return pipeline(
    partial(self.__parse, adapter),
    self.__shuntingYard,
    self.__calc)
```

```
def __shuntingYard(self, parsed):
    res = []
    stack = []
    for token in parsed:
      if token in self.OPERATORS:
        while stack and stack[-1] != "(" /
        and not self.__greaterPrecedence(token,stack[-1]):
          res.append(stack.pop())
        stack.append(token)
      elif token == ")":
        while stack:
          x = stack.pop()
          if x == "(":
            break
          res.append(x)
      elif token == "(":
        stack.append(token)
      else:
        res.append(token)
    while stack:
      res.append(stack.pop())
    return res
```

Тестирование

Использовалась очень мелкая выборка из корпуса документов – 100 документов. Результат сравнивался с поисковый движком ElasticSearch без учета ранжировая полученной выдачи.

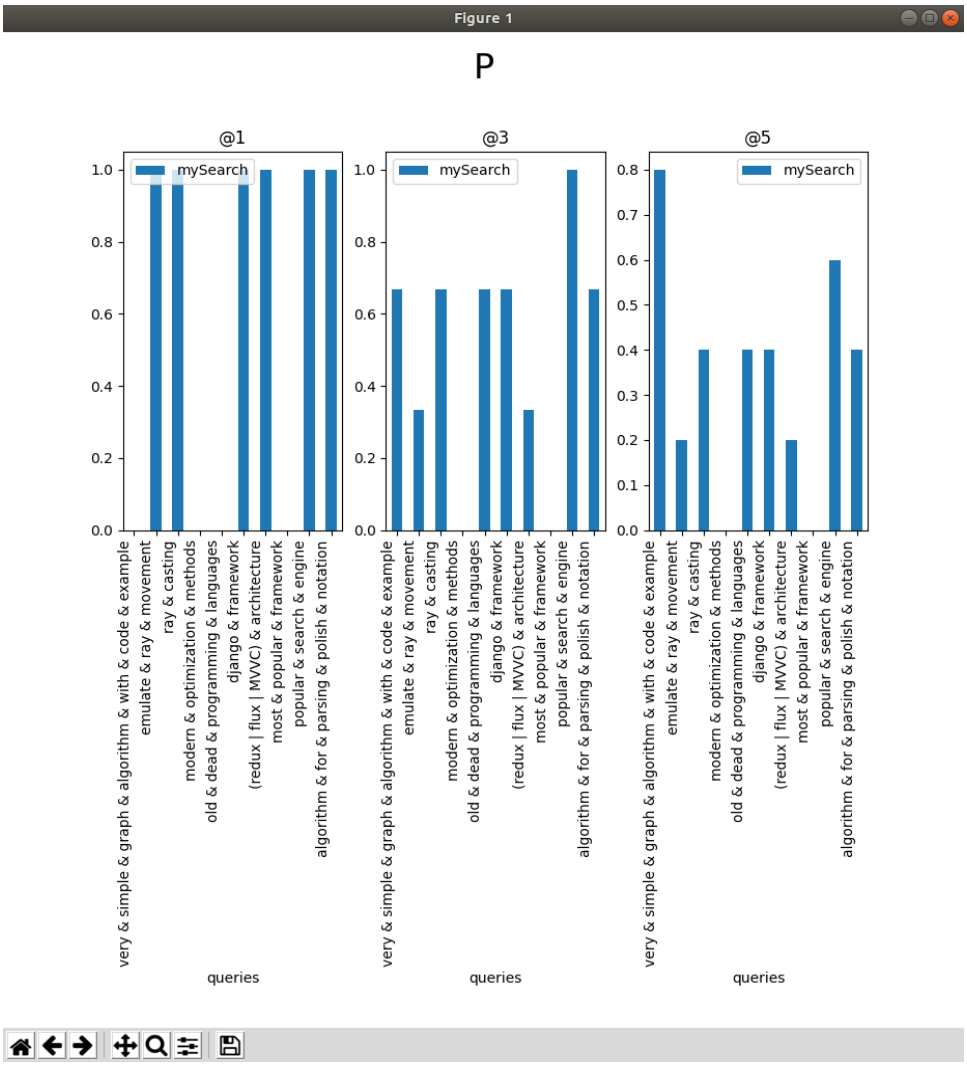
Далее приведены некоторые запросы, на которых проводились тесты:

python
new & programming & languages
django & framework
django & framework & !python
(redux | flux | MVVC) & architecture

Оценка поисковой выдачи

Сравнение проводилось на 10 запросах, которые были рассмотрены в ЛР2.

Оценки:



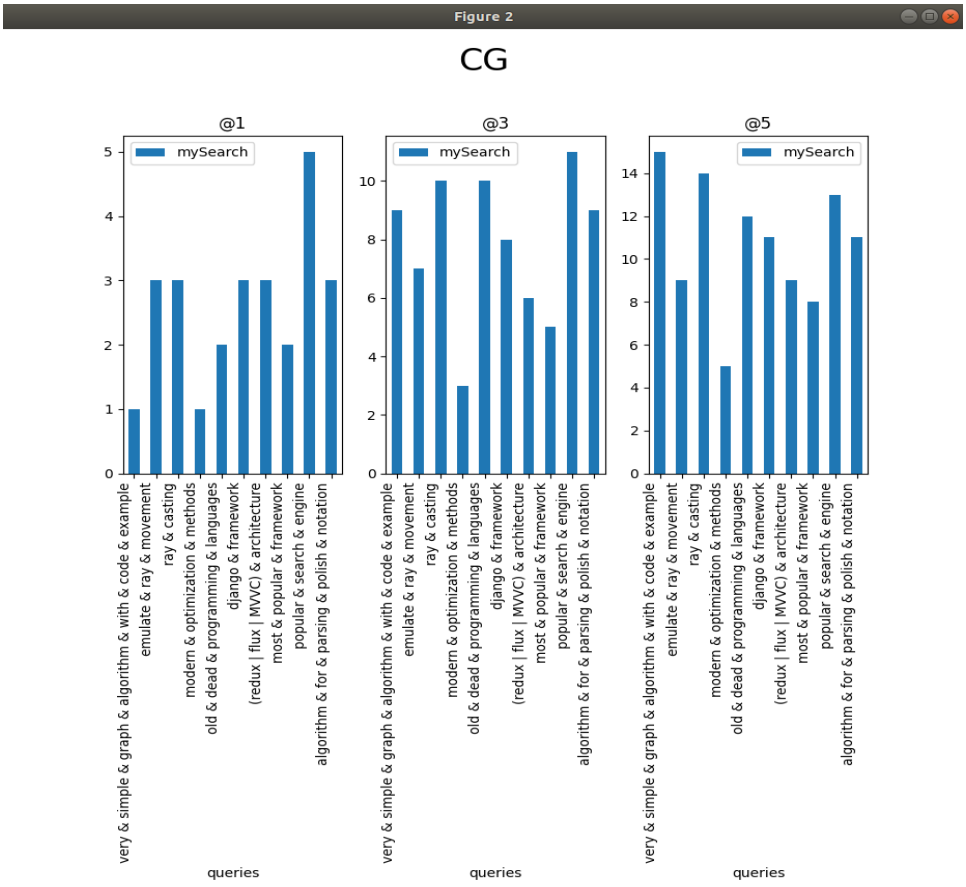


Figure 2
CG
@1
@3
@5
mySearch
very & simple & graph & algorithm & with & code & example
emulate & ray & movement
ray & casting
modern & optimization & methods
old & dead & programming & languages
django & framework
(redux | flux | MVVC) & architecture
most & popular & framework
popular & search & engine
algorithm & for & parsing & polish & notation
queries

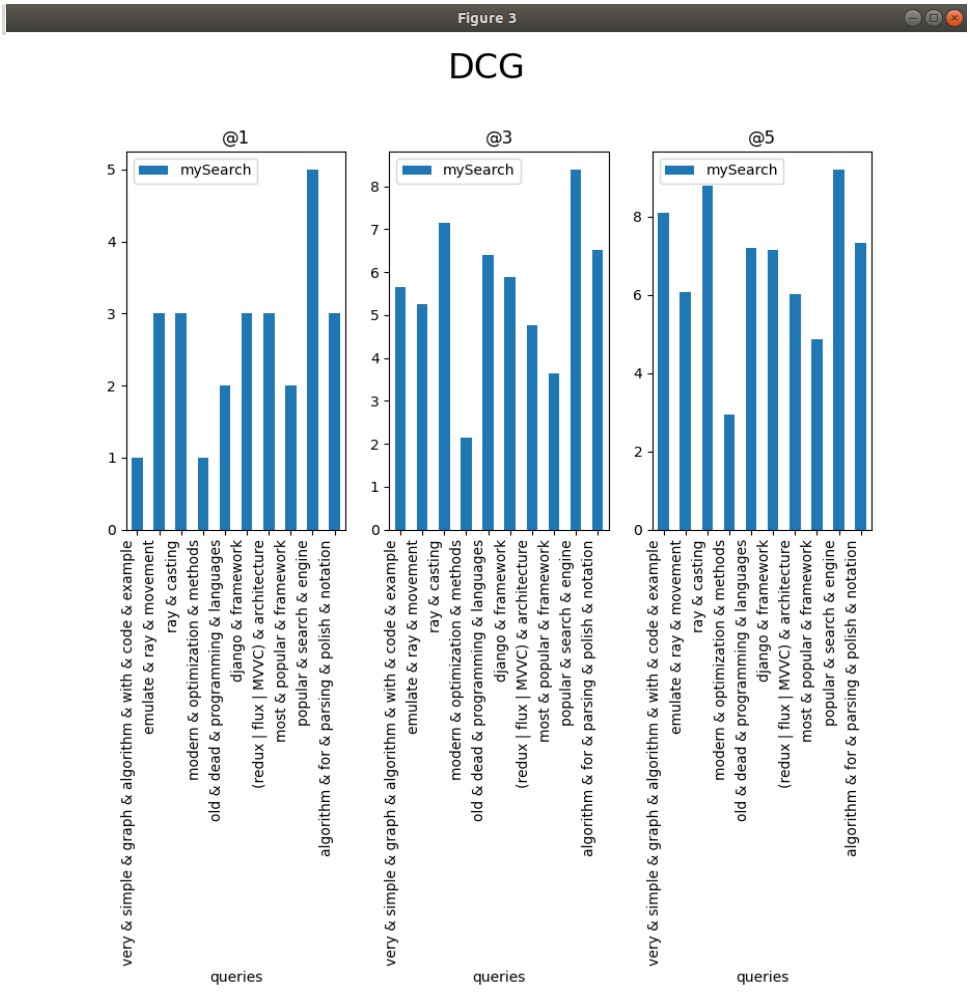


Figure 3
DCG
@1
@3
@5
mySearch
very & simple & graph & algorithm & with & code & example
emulate & ray & movement
ray & casting
modern & optimization & methods
old & dead & programming & languages
django & framework
(redux | flux | MVVC) & architecture
most & popular & framework
popular & search & engine
algorithm & for & parsing & polish & notation
queries
x= y=2.5164

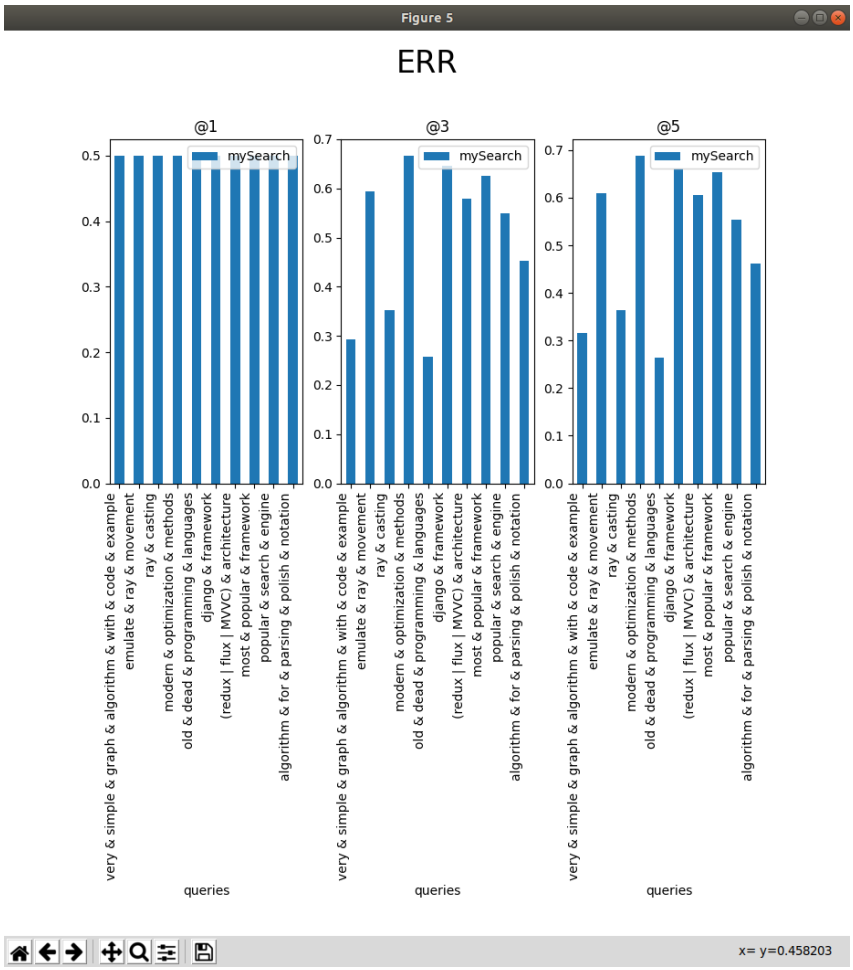


Figure 5
ERR
@1
@3
@5
mySearch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
very & simple & graph & algorithm & with & code & example
emulate & ray & movement
ray & casting
modern & optimization & methods
old & dead & programming & languages
django & framework
(redux | flux | MVVC) & architecture
most & popular & framework
popular & search & engine
algorithm & for & parsing & polish & notation
queries
x= y=0.458203

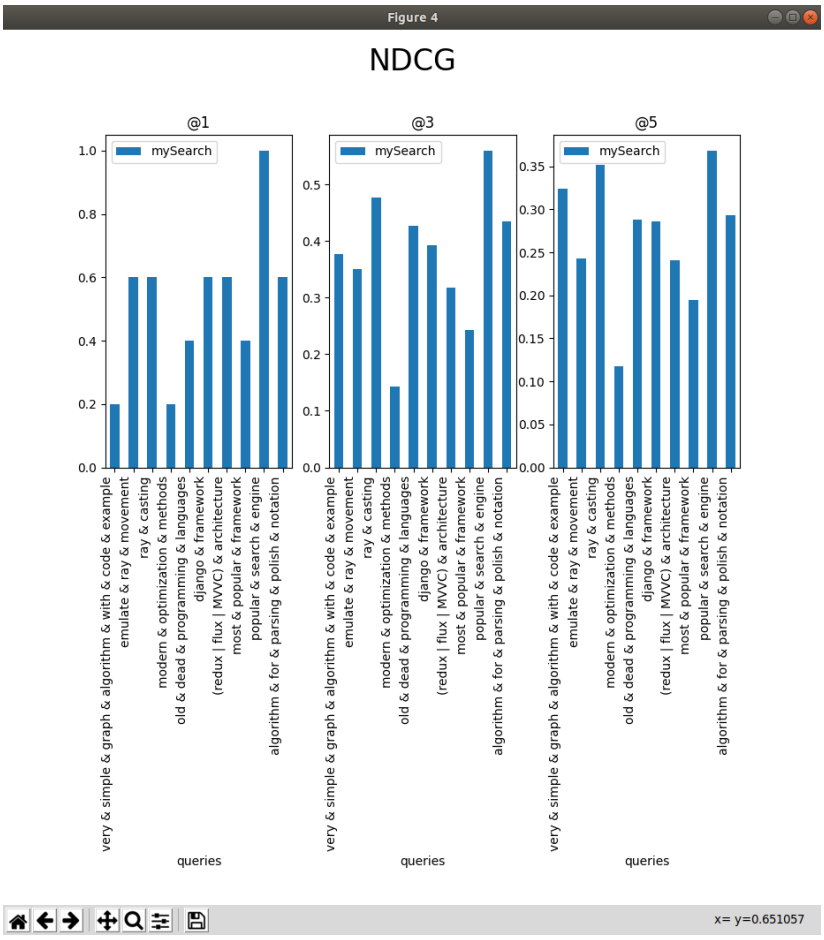


Figure 4
NDCG
@1
@3
@5
mySearch
queries
very & simple & graph & algorithm & with & code & example
emulate & ray & movement
ray & casting
modern & optimization & methods
old & dead & programming & languages
django & framework
(redux | flux | MVVC) & architecture
most & popular & framework
popular & search & engine
algorithm & for & parsing & polish & notation
x= y=0.651057

Оценки:

№	Запрос:	Источник:	SERP (5 первых)
1	python	Boolean	20437, 21038, 23076, 23527, 23836
2	very & simple & graph & algorithm & with & code & example	Boolean	157, 5003, 9818, 1182, 9044
3	where & to & apply & map & reduce	Boolean	9751, 10275, 10487, 8367, 4458
4	emulate & ray & movement	Boolean	13329, 13332
5	what & are & the & game & engines	Boolean	16011, 16450, 16142, 16009, 13357
6	ray & casting	Boolean	662, 680, 12436, 12207, 12454
7	faster & sorting & algorithm	Boolean	8001, 875, 8052, 926, 8098
8	modern & optimization & methods	Boolean	3886, 13061, 17313, 12729, 8876
9	new & programming & languages	Boolean	9135, 21013, 22266, 23704, 24567
10	old & dead & programming & languages	Boolean	23085, 24576, 24577, 24578, 24579
11	esoteric & programming & languages	Boolean	20362, 24036, 22488, 22127, 24767
12	django & framework	Boolean	16565, 21924, 21049, 19595, 18835
13	django & framework & !python	Boolean	16677, 15481, 16516, 16584, 16703
14	(redux | flux | MVVC) & architecture	Boolean	14380, 14379, 12013, 12173, 8901
15	SPA & frontend & frameworks	Boolean	14427, 16475, 20237, 16652
16	most & popular & framework	Boolean	21049, 21924, 16699, 20059, 13516
17	big & data & algorithms	Boolean	9809, 8286, 8455, 3238, 1504
18	popular & search & engine	Boolean	6584, 23576, 21188, 13516, 14146
19	n*log(n)	Boolean	0, 1, 2, 3, 4
20	algorithm & for & parsing & polish & notation	Boolean	11820, 8745, 8738, 24299, 20750

Анализ и недостатки

На некоторые запросы булев поиск не выдал результата, связать это можно с тем, что на данном этапе не производилась нормализация токенов.

На запросы булев поиск выдавал нерелевантные результаты на данном этапе не производится ранжирование, более лучший результат не попадает на более высокую позицию.

Для улучшения поиска следует ввести нормализацию токенов и производить ранжирование.

Вывод

В ходе выполнения лабораторной работы был реализован, протестирован и проанализирован булев поиск. Результаты выдачи, полученные данным поиском, были сравнены с результатами поиска ElasticSearch