

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Факультет Информационных Технологий и Управления
Кафедра ИТАС

Отчет по лабораторной работе №2
«ТЕХНОЛОГИЯ АНАЛИЗА ТЕКСТА И ИЗВЛЕЧЕНИЯ КЛЮЧЕВЫХ
СЛОВ»

Выполнил
студент группы
820601 Шведов А.Р

Проверил
Ярмолик В.И.

Минск, 2020

1 ЦЕЛЬ РАБОТЫ

Целью работы является практическое освоение технологии анализа текста, извлечения ключевых слов и профессионального поиска информации.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Знание общих принципов функционирования поисковых средств и умение грамотно составить запрос поисковой машине необходимые, но недостаточные условия успешного поиска требуемой информации. Надо еще и уметь правильно выбирать ключевые слова поиска. Это особенно ощутимо при поиске в незнакомой предметной области, поскольку выбор ключевых слов, которые должны максимально соответствовать тематическому направлению, затруднен именно незнанием специфики предметной области (заметим, что это штатная ситуация для поисковой машины). Выбор ключевых слов в данном случае может осуществить специалист узкого профиля, но труд его дорог и малопроизводителен, или специальные программные средства, основанные на применении законов Зипфа.

Джордж Зипф установил, что все тексты подчиняются общим закономерностям, и сформулировал в 1946—49 гг. несколько законов, которые нашли применение в технологии поиска информации.

Для ознакомления с положениями первого закона Зипфа введем, следуя Зипфу, необходимые терминологические определения. Рассмотрим некоторый произвольный текст. Выпишем все различающиеся слова данного текста в виде $s_1, s_2, \dots, s_1, \dots, s_1$, где s_i - i -ое слово, не совпадающее ни с каким другим словом в данном множестве. Для каждого из этих слов подсчитаем количество его повторов в тексте. В результате получим $f_1, f_2, \dots, f_1, \dots, f_1$, где f_i - количество повторений i -го слова в тексте, названное Зипфом **частотой слова** (в данном случае i -го слова).

Далее Зипф, располагая слова в порядке убывания их частот, поставил им в соответствие числа натурального ряда, назвав эти числа **рангами слов (R)**: слову с максимальной частотой присваивается ранг 1, следующему по частоте — ранг 2 и т.д. При этом если несколько разных слов имеют одинаковые частоты, то они объединяются в один блок. Наконец Зипф ввел понятие вероятности встречи слова как отношение частоты слова к общему количеству слов в тексте (в математической статистике такое отношение называется частотой события; а в справочной системе поисковых машин — относительной частотой).

Первый закон Зипфа утверждает, что произведение частоты встречи слова в тексте (или вероятности встречи слова по Зипфу) на его ранг есть величина приблизительно постоянная для любых текстов определенного языка, т.е. имеет место $C = f * R \approx Const$.

3 ИСПОЛЬЗУЕМЫЕ ПРОГРАММЫ

Возможность извлечения ключевых слов из текстовых материалов имеется и в текстовом редакторе *MS Word*, однако использование этой возможности дает неудовлетворительные результаты. В данной работе будут использоваться утилиты командной строки *bash*, такие как *bashfind*, *grep*, *awk*.

Awk – утилита предназначенная для простых, механических и вычислительных манипуляций над данными. Утилита изначально объединяла свойства утилит *UNIX* - *sed* и *grep*. В дальнейшем ее возможности значительно расширились. *Awk* "разбивает"каждую строку на отдельные поля. По-умолчанию, поля – это последовательности символов, отделенные друг от друга пробелами, однако имеется возможность назначения других символов, в качестве разделителя полей. Она анализирует и обрабатывает каждое поле в отдельности. Это делает его идеальным инструментом для работы со структурированными текстовыми файлами.

Утилита *tr* выполняет преобразование, подстановку (замену), сокращение и/или удаление символов, поступающих со стандартного ввода, записывая результат на стандартное устройство вывода. Она часто применяется для удаления управляющих символов из файла или преобразования регистра символов. Как правило, команде передаются две строки (набора) символов: первый набор СТРОКА1 содержит искомые символы, а второй СТРОКА2 - те, на которые их следует заменить. При запуске команды устанавливается соответствие между символами обоих наборов, а затем начинается преобразование.

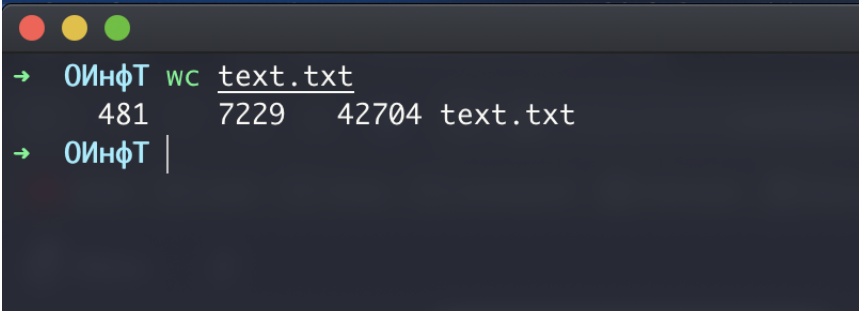
Для работы с текстом воспользуемся текстовым файлом, содержащим текст на английском языке в формате *.txt*.

4 ХОД РАБОТЫ

Проанализируем файл под названием *text.txt*. Посчитаем частоты вхождения слов и определим их ранги.

4.1 Базовый анализ файла

Посчитаем количество слов и строк в файле с помощью утилиты *wc* (*word count*). По умолчанию результат содержит 3 числа: количество строк, количество слов и количество байт в файле (рисунок 4.1).



```
→ ОИнфТ wc text.txt
      481      7229      42704 text.txt
→ ОИнфТ |
```

Рисунок 4.1 – Количество слов и строк в файле.

Посчитаем количество повторения разных слов. Для этого с помощью утилиты *tr* редактируем вывод из файла, заменив все пробелы между словами на знак новой строки. Это позволит нам с помощью *grep* найти все строки с нужным словом и перенаправить данные в утилиту *wc* (Рисунок 4.2). Используется флаг *-l* для того, чтобы посчитать только количество полученных от прошлой команды строк со словом, что и будет являться количеством слов в файле.

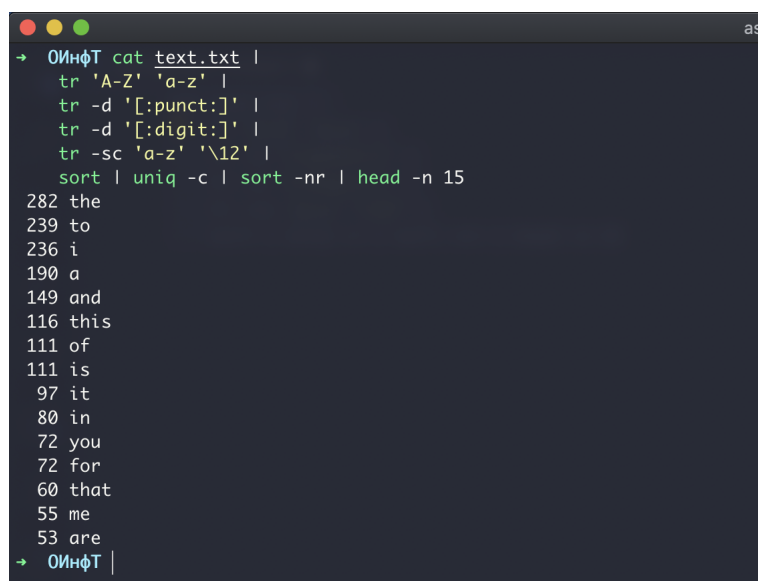


```
→ ОИнфТ tr ' ' '\n' < text.txt | grep "this" | wc -l
      85
→ ОИнфТ tr ' ' '\n' < text.txt | grep "code" | wc -l
      46
→ ОИнфТ tr ' ' '\n' < text.txt | grep "are" | wc -l
      76
→ ОИнфТ |
```

Рисунок 4.2 – Подсчет разных слов.

4.2 Обработка всего текста

Используя уже изученные возможности можно написать простой *shell*-скрипт для обработки документа. Он будет разбивать весь документ на набор строк из одного слова и считать количество строк согласно уникальности. Так же скрипт использует простое регулярное выражение, чтобы заменить все регистры на нижний. Он так же обрабатывает документ, удаляя все цифры и знаки препинания. Затем вывод команд будет сортироваться и показываться в формате Частота - Слово. Результат работы программы – Рисунок 4.3.



```
→ ОИнфТ cat text.txt |
  tr 'A-Z' 'a-z' |
  tr -d '[:punct:]' |
  tr -d '[:digit:]' |
  tr -sc 'a-z' '\12' |
  sort | uniq -c | sort -nr | head -n 15
282 the
239 to
236 i
190 a
149 and
116 this
111 of
111 is
97 it
80 in
72 you
72 for
60 that
55 me
53 are
→ ОИнфТ |
```

Рисунок 4.3 – Полный анализ текста

В результате обработки документа получено количество всех уникальных слов (не считая регистр), содержащееся в документе.

4.3 Создание поискового запроса

Так как проанализированный текст является художественно-разговорным, попробуем создать поисковой запрос согласно уникальным словам с малым рангом. Выведем 10 таких слов с помощью программы из п4.3 – Рисунок 4.4.

Воспользуемся поисковой системой *Google* – Рисунок 4.5

Так как поисковой запрос не показал искомого результата, выведем первые несколько строк файла (Рисунок 4.6) и попытаемся найти их (Рисунок 4.7).

```
→ ОИИФТ cat text.txt |
  tr 'A-Z' 'a-z' |
  tr -d '[:punct:]' |
  tr -d '[:digit:]' |
  tr -sc 'a-z' '\12' |
  sort | uniq -c | sort -nr | tail -n 15
1  adhoc
1  adapt
1  action
1  acquired
1  acquire
1  accountable
1  account
1  accidentally
1  abstracts
1  abstractions
1  absorbing
1  absorbed
1  absorb
1  abnormalities
1
→ ОИИФТ |
```

Рисунок 4.4 – Уникальные слова

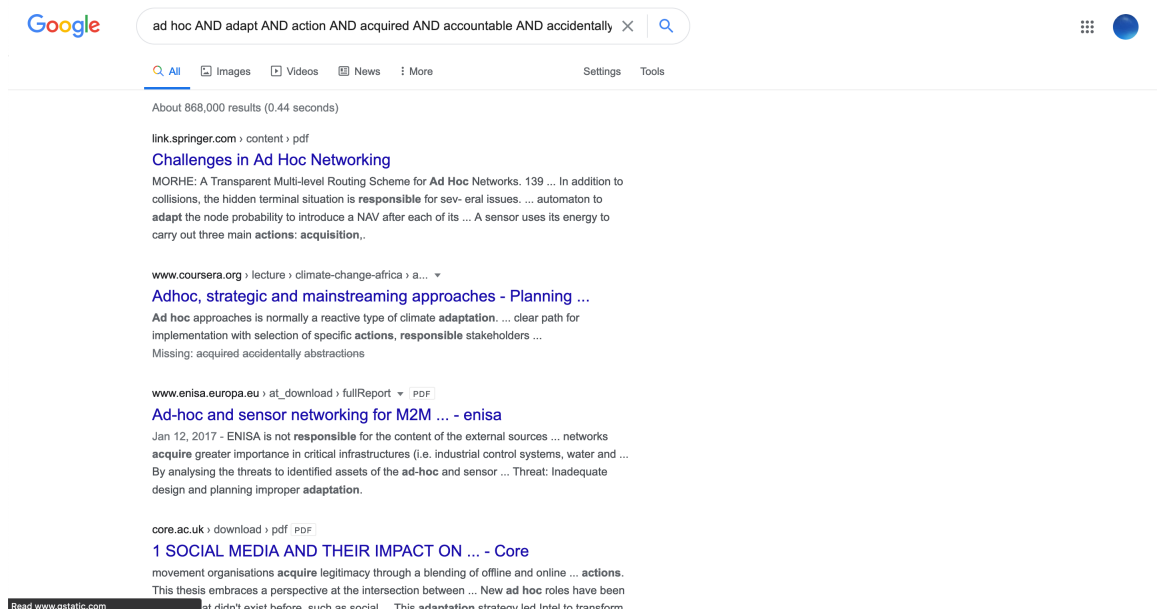
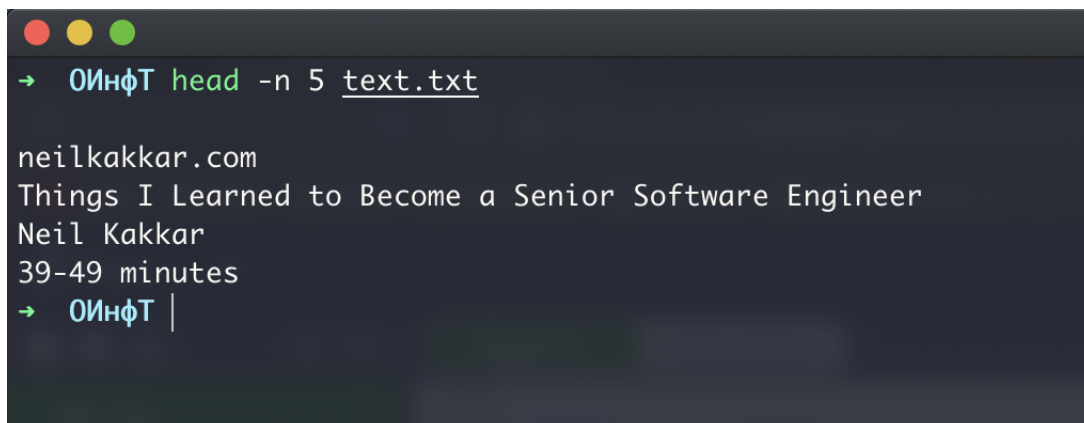


Рисунок 4.5 – Поиск по уникальным словам



```
→ ОИнфТ head -n 5 text.txt

neilkakkar.com
Things I Learned to Become a Senior Software Engineer
Neil Kakkar
39-49 minutes
→ ОИнфТ |
```

Рисунок 4.6 – Первые 5 строк файла.

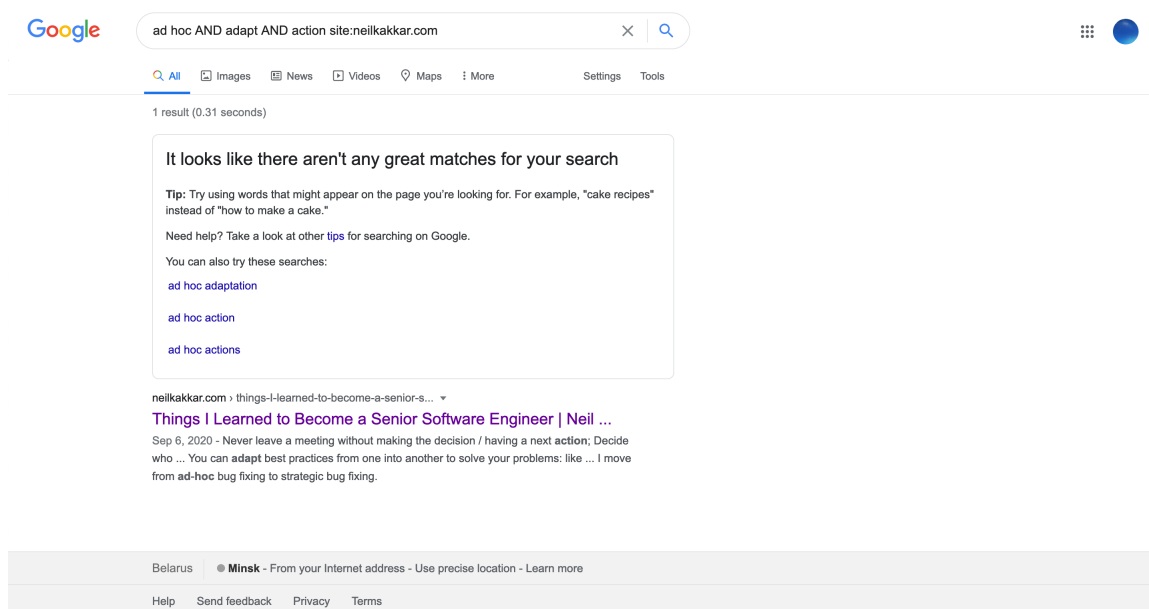


Рисунок 4.7 – Поиск исходного текста.

ЗАКЛЮЧЕНИЕ

Вывод: в ходе работы был обработан текстовый файл, получены ранги слов и были созданы поисковые запросы, которые проверились в поисковой машине *Google*.