МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ УКРАИНЫ
“КИЕВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ”
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра математических методов кибернетической безопасности

КУРСОВАЯ РАБОТА

Дисциплина: «Интеллектуальные методы обработки информации»

Направление подготовки: 8.04030101 «Прикладная математика»

Тема: «Интеллектуальные методы обработки информации»

Выполнил студент группы ФИ-51м

Кригин Валерий Михайлович

Проверила:

Бояринова Юлия Евгеньевна

_____

(подпись)

Оценка:

_____

Киев 2015

ОГЛАВЛЕНИЕ

1 ЗАКОН ЦИПФА

1.1 Закон Ципфа

Отношение ранга слова R, то есть его номер в списке слов, отсортированных по частоте в порядке убывания, к частоте слова f, является постоянным

$$Z = R \cdot f,$$

где f — частота слова в тексте, а Z — коэффициент Ципфа. Значит,

$$f = \frac{Z}{R}.$$

1.2 Задание

Под понятием “отфильтровать текст” тут и далее будут подразумеваться следующие действия:

1) очистить текст от всех символов кроме букв и пробелов;
2) буквы привести в нижний регистр, между словами оставить по одному пробелу.
3)

В лабораторной работе нужно

1) взять текст (желательно на русском языке) длиной более нескольких сотен килобайт;
2) отфильтровать текст;
3) составить частотный словарь слов — каждому слову текста сопоставить количество его повторений в тексте;

4) отсортировать частоты в порядке убывания;
5) изобразить полученные значения на графике, выбрав логарифмический масштаб для оси ординат и абсцисс;
6) построить степенную линию тренда и убедиться, что график похож на прямую линию, за исключением, возможно, “хвостов” с обеих концов.

1.3 Фильтр

На Perl написан фильтр, который
1) делает заглавные буквы строчными;
2) убирает всё кроме пробелов, символов табуляций, переносов строк и т.п.;
3) превращает все символы, которе не являются буквами, в пробел, также предотвращает появление двух пробелов подряд.

Вход считывается из stdin, выход происходит в stdout.

Листинг 1.1 — filter.pl

```
1 #!/usr/bin/perl -w -CAS
2 use utf8;
3
4 $_ = lc join('', <>);
5
6 s/[^\p{L}\s]//g;
7 s/[\s]+/ /g;
8
9 print;
```

1.4 Частотный словарь

На Python написан скрипт, который составляет частотный словарь и выводит его в формате csv. Полученный результат можно открыть в программе для работы с электронными таблицами для построения графиков.

Вход считывается из stdin, выход происходит в stdout.

Листинг 1.2 — counter.py

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  from sys import stdin
5  from os import linesep
6
7  words = ' '.join([l.strip() for l in stdin]).split(' ')
8
9  counts = {}
10 for key in set(words):
11     counts[key] = 0
12 for w in words:
13     counts[w] += 1
14
15 result = sorted(counts.iteritems(),key=lambda x: x[1],
16                  reverse=True)
17
18 print linesep.join('%s,%d'%r for r in result)
```

1.5 График

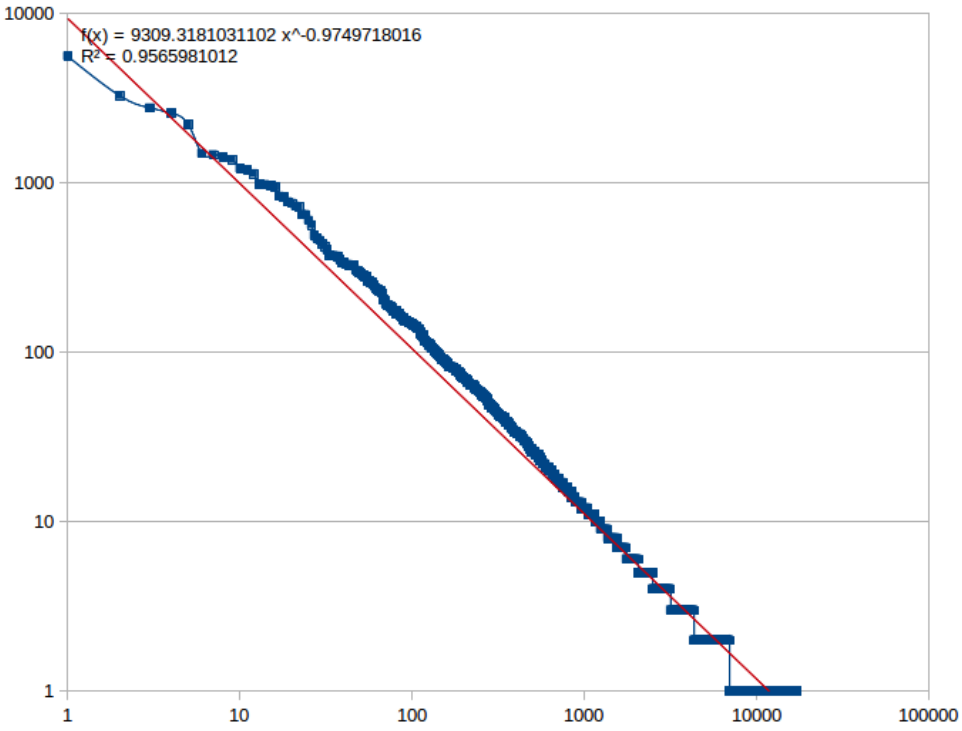

Рисунок 1.1 — Результат

2 ЗАКОН ХИПСА

2.1 Закон Хипса

Объём словаря уникальных слов $\nu(n)$ для текста длиной n связан с длиной текста следующим соотношением

$$\nu(n) = \alpha \cdot n^{\beta},$$

где α и β — эмпирические константы, которые разнятся от языка к языку, и для европейских языков колеблятся в пределах от 10 до 100 и от 0.4 до 0.6 соответственно.

2.2 Задание

В лабораторной работе нужно

1) взять текст (желательно на русском языке) длиной более нескольких сотен килобайт;
2) отфильтровать текст;
3) построить зависимость количества уникальных слов в тексте от его размера; для этого достаточно использовать один и тот же текст, изымать из него всё больше и больше слов с каждой итерацией, и подсчитывать число уникальных слов на каждом шаге;
4) изобразить полученные значения на графике;
5) построить степенную линию тренда и убедиться, что полученные параметры α и β близки к теоретическим значениям.

2.3 Фильтр

На Perl написан фильтр, который

1) делает заглавные буквы строчными;
2) убирает всё кроме пробелов, символов табуляций, переносов строк и т.п.;
3) превращает все символы, которе не являются буквами, в пробел, также предотвращает появление двух пробелов подряд.

Вход считывается из stdin, выход происходит в stdout.

Листинг 2.1 — filter.pl

```
1 #!/usr/bin/perl -w -CAS
2 use utf8;
3
4 $_ = lc join('', <>);
5
6 s/[^\p{L}\s]//g;
7 s/[\s]+/ /g;
8
9 print;
```

2.4 Частотный словарь

На Python написан скрипт, который считает зависимость между объёмом текста и объёмом словаря уникальных слов и выводит его в формате csv. Полученный результат можно открыть в программе для работы с электронными таблицами для построения графиков.

Листинг 2.2 — counter.py

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  from sys import stdin
5
6  words = ' '.join([l.strip() for l in stdin]).split(' ')
7
8  found = []
9
10 for i, w in enumerate(words):
11     if w not in found:
12         found.append(w)
13     print '%d,%d'%(i+1, len(found))
```

2.5 График

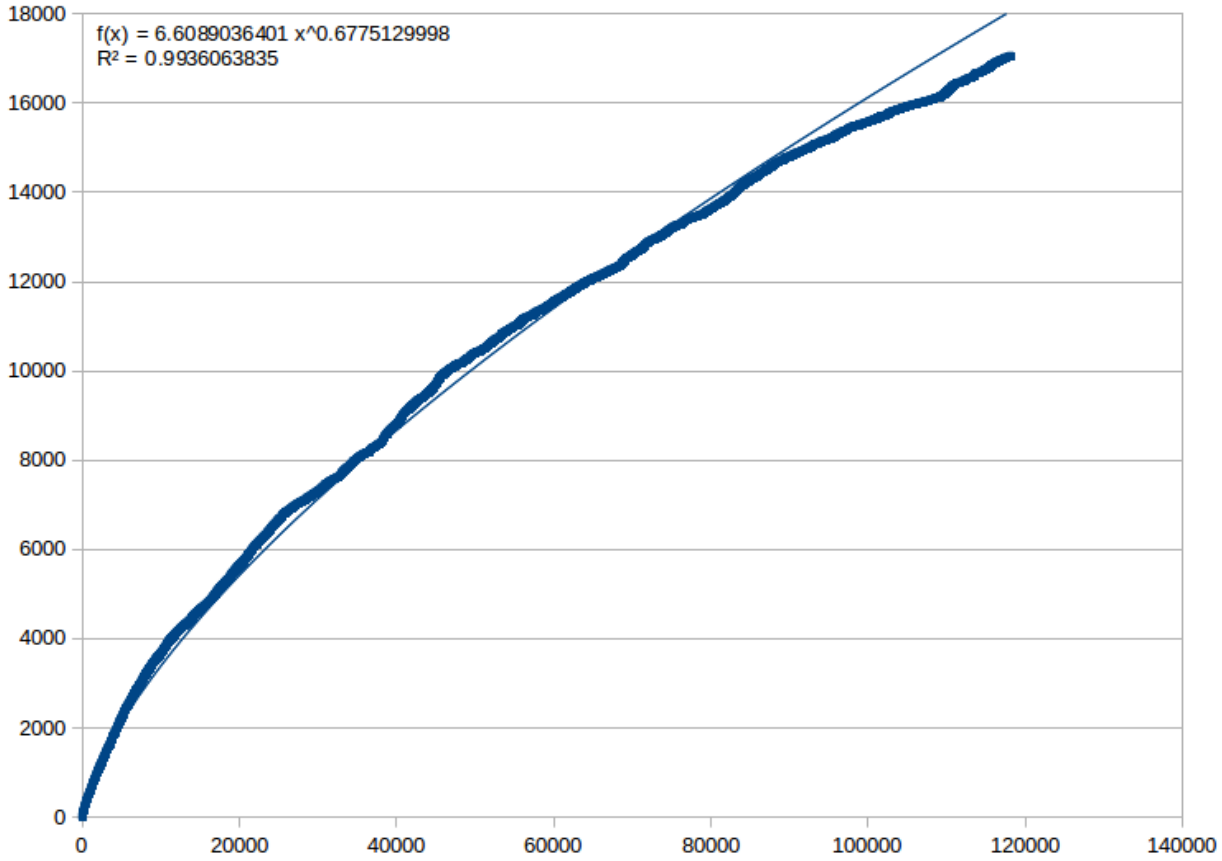

Рисунок 2.1 — Результат

3 $TF - IDF$

3.1 $TF - IDF$

Для i слова (n-граммы) индексы TF и IDF считаются по следующим формулам, где D — множество документов, n_k — количество повторений k слова (n-граммы) в текущем документе

$$TF_i = \frac{n_i}{\sum_k n_k},$$
$$IDF_i = \log \frac{|D|}{|\{d \mid t_i \in d \in D\}|}.$$

Сам индекс $TF - IDF$ является произведением индексов TF и IDF

$$TF - IDF_i = TF_i \cdot IDF_i$$

3.2 Задание

3.2.1 Основное задание

В лабораторной работе нужно

1) взять текст (желательно на русском языке) длиной более нескольких сотен килобайт;
2) отфильтровать текст;
3) подсчитать $TF - IDF$ для каждого слова;
4) изобразить полученные результаты в виде таблицы, отсортировав по значению $TF - IDF$ в порядке убывания.

То же самое нужно проделать с биграммами и триадами слов. Например,

в тексте “мама мыла раму” биграммы следующие: “мама мыла” и “мыла раму”.

3.3 Стоп-слова (шумовые слова)

Стоп-слова — те слова, которые не несут смысловую нагрузку. К ним относятся предлоги, частицы и прочее, если анализируемый документ не является учебником русского языка.

Список стоп-слов можно найти в интернете. Например, в разделе 12.9.4 Full-Text Stopwords документации к MySQL 5.5 находится список англоязычных шумовых слов.

Для увеличения скорости и уменьшения объёма обрабатываемых данных

1) при подсчёте $TF - IDF$ для слов можно выбросить из рассмотрения те, которые находятся в списке стоп-слов; например, слово “не” имеет мало смысла в сказке о царе Салтане, чего не скажешь о слове “лебедь”;
2) при подсчёте $TF - IDF$ для биграмм следует исключать те биграммы, которые содержат в себе шумовые слова; например, биграмма “я пришёл” имеет мало смысловой нагрузки, но биграмма “пришёл домой” скажет больше;
3) при подсчёте $TF - IDF$ для триад следует исключать те элементы, которые оканчиваются или начинаются на шумовые слова; скажем, “и она решила” мало о чём говорит, триада “она решила пойти” скажет больше, но “решила пойти домой” несёт определённый смысл.

3.4 Фильтр

На Perl написан фильтр, который

1) делает заглавные буквы строчными;

2) убирает всё кроме пробелов, символов табуляций, переносов строк и т.п.;
3) превращает все символы, которе не являются буквами, в пробел, также предотвращает появление двух пробелов подряд.

Вход считывается из stdin, выход происходит в stdout.

Листинг 3.1 — filter.pl

```
1 #!/usr/bin/perl -w -CAS
2 use utf8;
3
4 $_ = lc join('', <>);
5
6 s/[^\p{L}\s]//g;
7 s/[\s]+/ /g;
8
9 print;
```

3.5 Счётчик $TF - IDF$

На Python написан скрипт, который считает $TF-IDF$ для слов и выводит их в формате csv.

Полученный результат можно открыть в программе для работы с электронными таблицами для сортировки и фильтрации.

Листинг 3.2 — counter.py

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3
4 from sys import stdin, argv
```

```
 5 from os import linesep
 6 from math import log
 7
 8
 9 def get_count(words):
10     tfs = {}
11     for key in set(words):
12         tfs[key] = 0
13     for w in words:
14         tfs[w] += 1
15     return tfs
16
17 def group_n_grams(words, n):
18     if n < 2:
19         return words
20     return ['␣'.join(w for w in words[i:i+n]) for i in range(len
21
22 if __name__ == '__main__':
23     n_grams_length = 1
24     if len(argv) > 1:
25         n_grams_length = int(argv[1])
26
27     texts = ([l.strip().split('␣') for l in stdin])
28     names = map(lambda text: text[0], texts)
29     texts = map(lambda text: group_n_grams(text[1:], n_grams_len
30
31
```

```
32      tfs = map(get_count, texts)
33
34
35      idf = {}
36      for word in set(sum(texts, [])):
37          idf[word] = 0
38
39
40      for tf in tfs:
41          for word in tf:
42              idf[word] += 1
43
44
45      logN = log(len(texts))
46      for word in idf:
47          idf[word] = logN - log(idf[word])
48
49
50      tf_idfs = []
51      for i, tf in enumerate(tfs):
52          tf_idfs.append({})
53          for word in tf:
54              tf_idfs[i][word] = tf[word] * idf[word] / len(tf)
55
56      result = [(names[i], word, tf_idf[word], idf[word]) for i, t
57      result = sorted(result, key=lambda x: x[3], reverse=True)
58      print linesep.join('%s,%s,%f,%f'%(r) for r in result)
```

3.6 Результат

На рисунке 3.1 изображены первые 40 строк таблицы со значениями $TF - IDF$ для слов из 144 документов автора Льва Николаевича Толстого, 27 документов Фёдора Михайловича Достоевского и 31 документа Александра Сергеевича Пушкина, отсортированных по значению $TF - IDF$ в порядке убывания.

Объём документов Толстого 18MB, Достоевского 7.6MB, Пушкина — 2.8MB. Фильтрация происходит соответственно 10.3, 3.2 и 2 секунды. Далее каждый документ имеет только один перенос строки, который говорит об окончании документа, и их можно объединить в один файл. Подсчёт $TF - IDF$ происходит за 6.5 секунд, на выходе получается *.csv* файл объёмом 39MB.

Pushkin_Aleksandr__Kamennyi_gost	гуан	0.500417
Tolstoi_Lev__Vorobei	воробей	0.420966
Tolstoi_Lev__Ech_i_zayac	ёж	0.390793
Tolstoi_Lev__Filipok	филипок	0.38941
Tolstoi_Lev__Volga_i_Vazuza	волга	0.378673
Pushkin_Aleksandr__Kamennyi_gost	дон	0.37697
Tolstoi_Lev__Pesnya_pro_srachenie_na_reke_Chernoi	bis	0.375332
Tolstoi_Lev__Telenok_na_ldu	телёнок	0.366087
Tolstoi_Lev__Volga_i_Vazuza	вазуза	0.350113
Tolstoi_Lev__Vorobei	лён	0.331767
Tolstoi_Lev__Shat_i_Don	шат	0.313115
Tolstoi_Lev__Letuchaya_mysh	летучая	0.287007
Tolstoi_Lev__Zaicy_i_lyagushki	зайцы	0.270501
Tolstoi_Lev__Shakaly_i_slon	слон	0.270481
Pushkin_Aleksandr__Mocart_i_Saleri	моцарт	0.253782
Tolstoi_Lev__Proezchii_i_krestyanin	проезжий	0.251605
Tolstoi_Lev__Krestnik	крестник	0.249824
Tolstoi_Lev__Gde_lubov_tam_i_bog	авдеич	0.24433
Tolstoi_Lev__Bednye_ludi	жанна	0.241784
Tolstoi_Lev__Shat_i_Don	дон	0.240161
Tolstoi_Lev__Assiriiskii_car_Asarhadon	лаилиэ	0.238039
Tolstoi_Lev__Letuchaya_mysh	мышь	0.225394
Tolstoi_Lev__Kavkazskii_plennik	жилин	0.218334
Dostoevskii_Fedor__Slaboe_serdce	вася	0.217604
Tolstoi_Lev__Mnogo_li_cheloveku_zemli_nuchno	пахом	0.217452
Tolstoi_Lev__Rabotnik_Emelyan_i_pustoi_baraban	емельян	0.211203
Tolstoi_Lev__Zaicy_i_lyagushki	лягушки	0.206914
Pushkin_Aleksandr__Mocart_i_Saleri	сальери	0.202256
Tolstoi_Lev__Tri_starca	архиерей	0.200278
Tolstoi_Lev__Shakaly_i_slon	шакал	0.198928
Tolstoi_Lev__Proezchii_i_krestyanin	крестьянин	0.194574
Tolstoi_Lev__Myshi	кота	0.189198
Dostoevskii_Fedor__Dvoinik	голядкин	0.184289
Tolstoi_Lev__Tri_vora	козу	0.184254
Tolstoi_Lev__Shakaly_i_slon	шакалы	0.181451
Pushkin_Aleksandr__Skupoi_rycar	альбер	0.18117
Pushkin_Aleksandr__Kamennyi_gost	дона	0.177993
Tolstoi_Lev__Ech_i_zayac	заяц	0.174024
Tolstoi_Lev__Sobaka_i_ee_ten	собака	0.173506
Tolstoi_Lev__Zerno_s_kurinoe_yaico	зерно	0.169045

Рисунок 3.1 — Результат