

**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 . . . . .	3
1.1 Закон Ципфа . . . . .	3
1.2 Задание . . . . .	3
1.3 Фильтр . . . . .	4
1.4 Частотный словарь . . . . .	5
1.5 График . . . . .	6
2 Закон Хипса . . . . .	7
2.1 Закон Хипса . . . . .	7
2.2 Задание . . . . .	7
2.3 Фильтр . . . . .	8
2.4 Частотный словарь . . . . .	8
2.5 График . . . . .	9
3 $TF - IDF$ . . . . .	11
3.1 $TF - IDF$ . . . . .	11
3.2 Задание . . . . .	11
3.2.1 Основное задание . . . . .	11
3.2.2 Стоп-слова (шумовые слова) . . . . .	12
3.3 Фильтр . . . . .	12
3.4 Счётчик $TF - IDF$ . . . . .	13
3.5 Результат . . . . .	16

# 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, которые 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

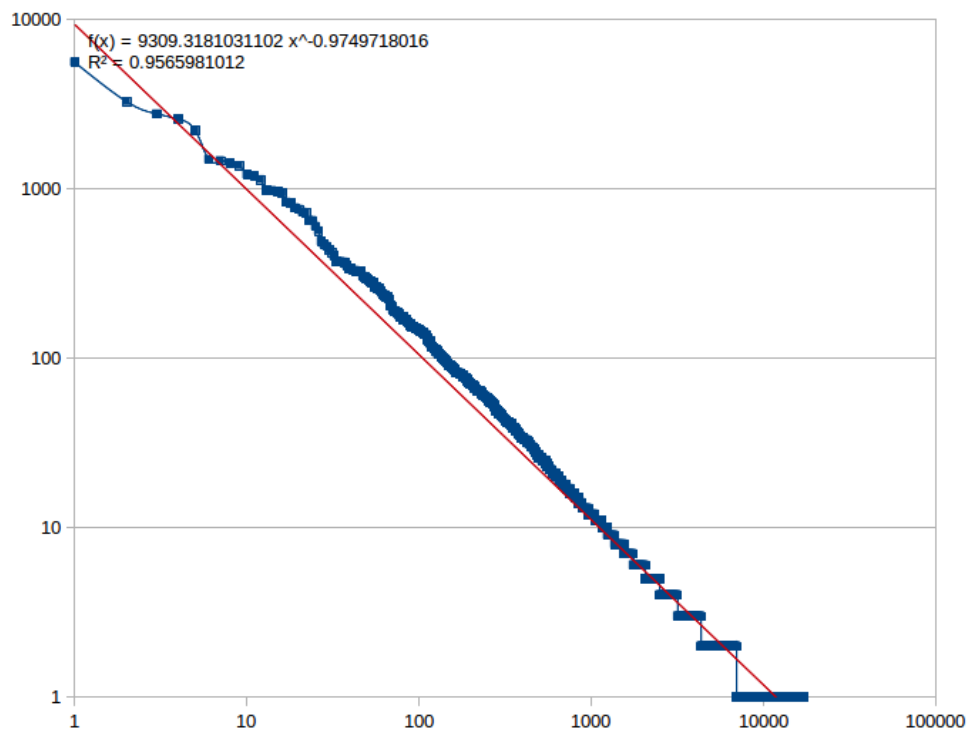


Рисунок 1.1 — Результат

## 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  $\alpha$  и  $\beta$  — эмпирические константы, которые разнятся от языка к языку, и для европейских языков колеблются 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  $\alpha$  и  $\beta$  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, которые 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

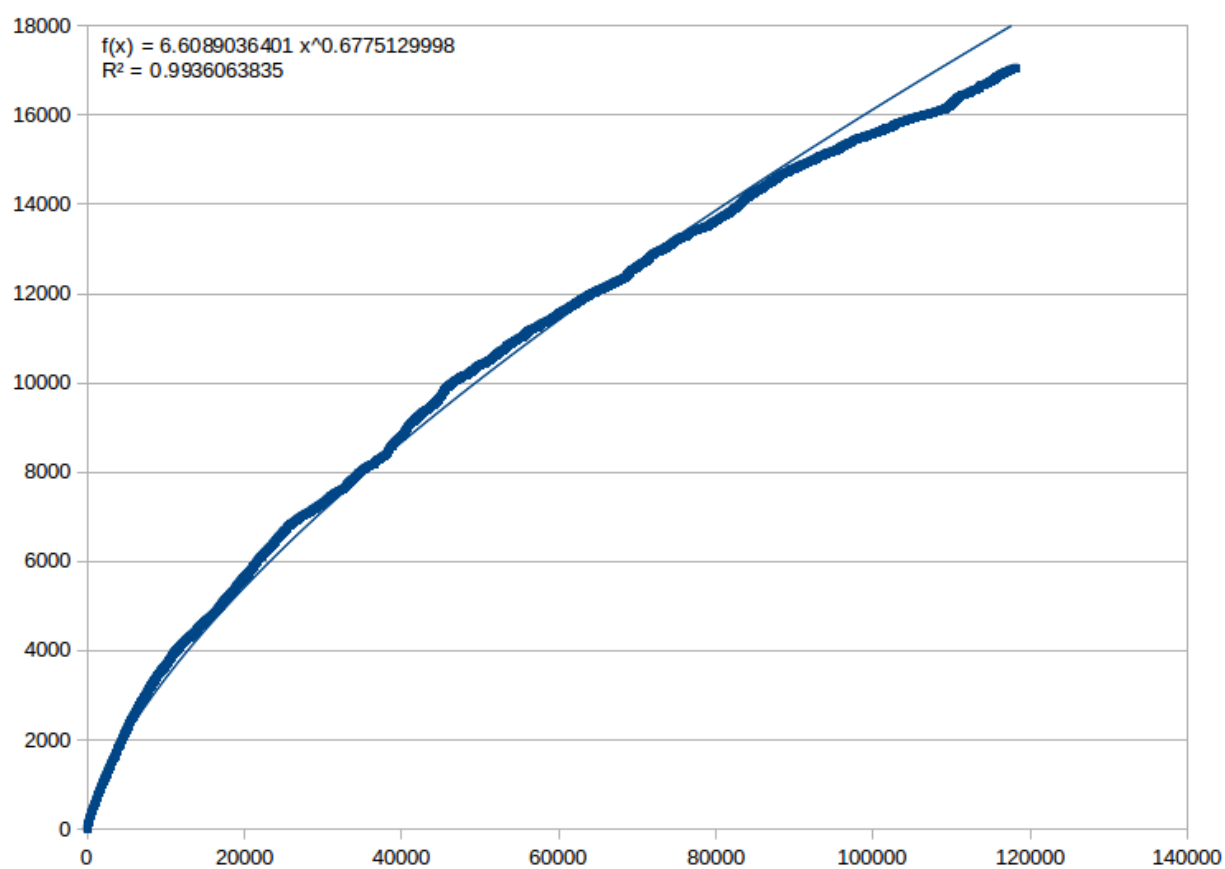


Рисунок 2.1 — Результат

### 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”.

### 3.2.2 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.

### 3.3 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, которые 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;
```

### 3.4 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 from stoplist import stop_list
8
9
10 def get_count(words):
11     tfs = {}
12     for key in set(words):
13         tfs[key] = 0
14     for w in words:
15         tfs[w] += 1
16     return tfs
17
18 def group_n_grams(words, n):
19     if n < 2:
20         return [w for w in words if w not in stop_list]
21     return ['_'.join(w for w in words[i:i+n])
22            for i in range(len(words)-n)
23            if words[i+n-1] not in stop_list
24            and words[i] not in stop_list]
25
26 if __name__ == '__main__':
27     n_grams_length = 1
28
29     if len(argv) > 1:
30         n_grams_length = int(argv[1])
31

```

```

32     texts = ([l.strip().split('_') for l in stdin])
33     names = map(lambda text: text[0], texts)
34     texts = map(lambda text: group_n_grams(text[1:],
35                                             n_grams_length), texts)
36
37
38     tfs = map(get_count, texts)
39
40
41     idf = {}
42     for word in set(sum(texts, [])):
43         idf[word] = 0
44
45
46     for tf in tfs:
47         for word in tf:
48             idf[word] += 1
49
50
51     logN = log(len(texts))
52     for word in idf:
53         idf[word] = logN - log(idf[word])
54
55
56     tf_idfs = []
57     for i, tf in enumerate(tfs):
58         tf_idfs.append({})

```

```

59         for word in tf:
60             tf_idfs[i][word] = tf[word] * idf[word] / len(tf)
61
62     result = [(names[i], word, tf_idf[word])
63               for i, tf_idf in enumerate(tf_idfs)
64               for word in tf_idf]
65     result = sorted(result, key=lambda x: x[2], reverse=True)
66     print linesep.join('%s,%s,%f'%(r) for r in result)

```

### 3.5 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 18МВ, Достоевского 7.6МВ, Пушкина — 2.8МВ. 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 39МВ.



№	Книга	Слово	$TF - IDF$
1	TolstoiVorobei	воробей	0.910196
2	TolstoiEchizayac	ёж	0.723855
3	TolstoiVorobei	лён	0.717333
4	TolstoiTelenoknaldu	телёнок	0.649992
5	TolstoiLetuchayamysh	летучая	0.645765
6	PushkinKamennyigost	гуан	0.625634
7	TolstoiVolgaiVazuza	волга	0.616940
8	TolstoiFilipok	филипок	0.603935
9	TolstoiShatiDon	шат	0.591983
10	TolstoiShakalyislon	слон	0.570469
11	TolstoiVolgaiVazuza	вазуза	0.570408
12	TolstoiPesnyaprosrachenienarekeChernoi	bis	0.523350
13	TolstoiZaicyilyagushki	зайцы	0.517134
14	TolstoiLetuchayamysh	мышь	0.507136
15	PushkinKamennyigost	дон	0.471297
16	TolstoiMyshi	кота	0.467739
17	TolstoiShatiDon	дон	0.454054
18	TolstoiSobakaieeten	собака	0.441651
19	TolstoiShakalyislon	шакал	0.419556
20	TolstoiZaicyilyagushki	лягушки	0.395571
21	TolstoiShakalyislon	шакалы	0.382696
22	TolstoiSobakaieeten	своё	0.356543

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_Idu	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_Dvoynik	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 для слов