Дальневосточный математический журнал. 2020. Т. 20. № 2. С. 155–163

УДК 519.76
MSC2010 62-04

© М. А. Гузев[1], М. А. Князева[2], И. И. Москалев[2], Е. Ю. Никитина[2]

Ранговый анализ компьютерных программ

В данной работе описано применение метода рангового анализа для формальных языков на примере исходных кодов программ на языке Java. При описании технологии составления частотных словарей внимание акцентировано на том, как учитывается наличие анафорики в компьютерных программах. При обсуждении результатов отмечено, что структурным компонентам программ различного назначения соответствуют определенные области графика зависимости ранга от частоты встречаемости лексемы.

Ключевые слова: *ранговые распределения, закон Ципфа, частотные словари, формальные языки.*

DOI: https://doi.org/10.47910/FEMJ202017

Введение

Существуют различные подходы в исследовании лингвистических закономерностей в языках коммуникаций как человеческих, так и машинных. Традиционно изучением текстов различного рода занимается лингвистика, которая в широком смысле может пониматься как часть семиотики — науки о знаках и знаковых системах. Текст представляется как совокупность слов (знаков) и всех их производных (словоформ) из определенного заранее словаря. К настоящему времени для количественного исследования знаковых систем широко применяется метод ранговых распределений, использующий функции, отображающие отрезок натурального ряда чисел в множество неотрицательных вещественных чисел [1, с. 9-20]. Применительно к текстовым данным ранговое распределение трактуется как упорядоченная в порядке убывания по частоте появления совокупность наименований элементов, то есть частота является функцией номера элемента в словаре, составленном из различных слов и словоформ.

В предшествующих работах [2, 3, с. 117-129, с. 180-190] дана подробная историческая справка о развитии понимания природы ранговых распределений. Приведем здесь краткие выдержки из них.

[1] Институт прикладной математики ДВО РАН, 690041, г. Владивосток, ул. Радио, 7.
[2] Дальневосточный федеральный университет, 690922, г. Владивосток, о. Русский, п. Аякс, 10.
Электронная почта: guzev@iam.dvo.ru (М. А. Гузев), nikitina.eyu@dvfu.ru (Е. Ю. Никитина).

Впервые ранговое распределение получено в 1913 году немецким ученым-физиком Феликсом Ауэрбахом. На основе обширного фактического материала он выявил закон концентрации населения:

$$P_N = \frac{P_1}{N}, \tag{1}$$

где P_N — численность населения города N-го ранга; P_1 — численность населения города 1-го ранга в иерархии городских поселений (город с максимальным количеством жителей) (см., например, [4]). Закон Ауэрбаха не получил широкой известности, однако вскоре закономерности поведения характеристик явлений и объектов, формализуемые с помощью соотношения (1), были вновь найдены в других областях человеческой деятельности.

В 1949 году гарвардский профессор филологии Джордж Кингсли Ципф сформулировал принцип наименьших усилий, который фактически являлся вновь открытым принципом Парето (1897 г.): ресурсы (люди, товары, время, знания или любой другой источник продукта) самоорганизуются так, чтобы свести к минимуму затраченную работу. При этом в пределах 20-30% любого ресурса достигается 70-80% результатов деятельности, выполняемой на основе этого ресурса [5].

Дж. К. Ципф развил этот постулат, предположив, что устойчивый равновесный характер принципа наименьших усилий складывается из компромисса между двумя противоположными тенденциями во взаимоотношении между отдельным индивидом и обществом в целом. В коммуникативной практике это означает, что говорящий стремится истратить как можно меньше слов и быть понятым, а слушающие требуют как можно более разнообразного текста, чтобы облегчить себе его понимание [6].

Ципф экспериментально показал, исследовав распределения слов во многих текстовых документах, что если для какого-нибудь довольно большого текста составить список всех слов, которые встретились в нем, а потом ранжировать эти слова в порядке убывания частоты их появления в тексте, то для любого слова произведение его ранга и частоты появления будет величиной постоянной:

$$\omega r = C, \tag{2}$$

где ω — частота встречаемости слова в тексте; r — ранг слова в списке слов; C — эмпирическая постоянная величина (коэффициент Ципфа).

Распределение (2) показывает, что человек, исходя из принципа наименьших усилий, строит текст из часто повторяемых им слов (имеющих небольшие ранги в словаре), и не тратит силы на поиск редко употребляемых, с большими рангами. Говоря современными цифровыми терминами, происходит обращение к оперативной памяти, а не к долговременной.

Еще одно проявление цифровизации выражено в том, что при кодировании (оцифровке) текста все знаки приобретают некую «стоимость», определяемую частотой появления слова в тексте. Следовательно, закон Ципфа можно объяснить тем, что, создавая текстовое произведение, человек интуитивно стремится к оптимальному (с точки зрения кодирования) представлению информации — чтобы каждый символ (знак) переносил максимальное количество информации.

В 1952 г. закон Ципфа был развит Бенуа Мандельбротом, соединившим теорию информации, психолингвистику и теорию вероятностей. Исследуя задачу оптимального кодирования, Мандельброт уточнил формулу Ципфа (2), введя в нее дополнительные параметры. Это позволило приблизить данную зависимость к реально наблюдаемым данным. Закон Ципфа – Мандельброта постулирует, что, исходя из требований минимальной стоимости сообщений, частота появления знака в сообщении ω обратно пропорциональна его рангу r в некоторой степени γ для словаря:

$$\omega = \frac{C}{r^{\gamma}}, \tag{3}$$

где γ — величина (близкая к единице), которая зависит от свойств текста [7]. Справедливость выполнения закона для исследуемого текста соответствует его некоторой сбалансированности и системности [1]. В [8, с. 384] показано, что стабильная, сбалансированная лингвистическая система имеет в ранговом распределении (3) показатель степени γ в пределах от 0,5 до 1,5.

Закон Ципфа – Мандельброта определяет эмпирическую закономерность распределения частоты слов естественного языка. В данной работе метод ранговых распределений используется для анализа лингвистических свойств компьютерных программ, а именно для выявления значимых параметров программного кода для семейства схожих по функциональности программ.

1. Использование подхода В. П. Маслова для анализа исходных кодов программных средств

При решении задачи потребуется построение соответствующих эмпирических зависимостей частоты от ранга и подбор аппроксимирующих функций. В связи с этим следует указать предложенный В. П. Масловым подход, описывающий ситуацию общего положения. Он состоит в том, что параметры моделирования зависимости между частотой встречаемости слова и рангом слова являются характеристикой языка автора и точнее, чем закон Ципфа, описывают соотношения между частотой встречаемости слов и другими параметрами словаря. Пусть составлен словарь, в котором указаны частоты встречаемости каждого слова некоторого объемного текста. Если выбирать в нём слова случайным образом, то какова вероятность попасть на слово с заданной частотой? При таком рассмотрении частота ω_i выступает в роли случайной величины, а число слов n_i с этой частотой — в роли числа выпадений этой случайной величины. Следует отметить, что язык предоставляет возможность эффективно кодировать сообщения за счет использования слов-заместителей (местоимений) и пропуска легко подразумеваемого слова (эллипсис). Носители языка легко справляются с задачей анафорики — восстановления заменяемых слов и эллипсиса. Проведя процедуру анафорики, можно точно подсчитать длину восстановленного текста и сравнить реальные и восстановленные (виртуальные) частоты одних и тех же слов. Очевидно, что в виртуальном тексте частота встречаемости слов увеличивается пропорционально исходной частоте. Такое представление, предложенное В. П. Масловым [9, с. 315-325]– [13], позволяет получить более точное соотношение

между рангом слова r и частотой ω_i его встречаемости в словаре:

$$r = \sum_{i=1}^{l} \frac{1}{e^{\beta\omega_i+\sigma} - 1}.$$

Практическое применение полученных соотношений связано с выбором виртуальной частоты и других феноменологических параметров. Простейшая параметризация для виртуальной частоты встречаемости имеет вид $\widetilde{\omega}_i = \omega_i(1+\alpha\omega_i^{\gamma}), \alpha > 0, \gamma > 0$. Полагая $\beta << 1$ и $\sigma = 0$, можно перейти от суммы к интегралу, в результате имеем

$$r \cong ln\frac{\omega^{\gamma}}{1+\alpha\omega^{\gamma}} + c. \tag{4}$$

В своих работах В. П. Маслов показал, что формула (4) точнее, чем закон Ципфа, описывает соотношение между рангом слова и частотой его встречаемости в словаре. Объективность модели неоднократно использована авторами при проведении исследований в различных областях, в том числе слабоформализованных (например, [2, 3, 14, с. 117-129, с. 180-190, с. 110-123]). Поэтому еще одно расширение метода В.П.Маслова на высокоуровневые формальные языки компьютерных программ можно выполнить довольно естественно. Данную модель (4) будем использовать при построении и исследовании частотных словарей исходных текстов компьютерных программ. Авторы предполагают, что такое построение может раскрыть новые возможности для определения объективных характеристик и классификации компьютерных текстов.

Высокоуровневый язык программирования — это знаковая система, предназначенная для коммуникации человека и вычислительной машины. Как и в любом естественном, в этом языке имеется набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — компьютер) под её управлением. Основная структурная единица языка — лексема. Все лексемы формального языка имеют определенный смысл и назначение: зарезервированные ключевые слова, имеющие однозначное толкование в программе и являющиеся управляющими для процессора, идентификаторы, которые придумывает программист для хранения и доступа к данным, либо для описания создаваемых им типов и объектов; служебные слова для вспомогательных нужд. Весь этот набор лексем можно рассматривать как словарь высокоуровневого языка программирования. Каждая программа, написанная на этом языке, будет содержать как набор стандартных лексем служебного и вспомогательного назначения, так и созданные самим программистом уникальные лексемы.

В коммуникативной системе есть понятия генератора и потребителя. Генератором текстов программ является программист, потребителем — транслятор, который переводит текст программы с высокоуровневого языка программирования на язык машинных команд, исполняемых процессором.

Характер коммуникации «человек — машина» накладывает следующие ограничения:

– конечное число слов, значение которых понятно транслятору (ключевые слова);

– строгие правила записи команд, от которых запрещено отступать.

Тем не менее в компьютерных текстах используются явления, заимствованные из практики владения естественными языками, облегчающие авторам создание текстов программ — например, анафорика. Необходимо уточнить, что в программировании обработка анафорики — неотъемлемая рутина любого компилятора, и процедура восстановления заменяемых местоимениями слов является необходимым промежуточным этапом анализа текста программы. Например, для процедурно-ориентированного языка снятие анафорики продемонстрируем на примере программы, вычисляющей $n!$ (Рис. 1).

Имеем описание функции вычисления n!

```
int factorial(int n)
{
   if (n < 1) {
         printf("Error\n");
         break;
   }
   else {
         for (int i = n-1; i > 1; i--)
    {          n *=i;
    }
   }
return n;
 }
```

Встретив в тексте программы команду

```
Y=factorial(m);
```

при разборе компилятор заменяет это выражение на следующий текст (формальный параметр n становится фактическим параметром m):

```
if (m < 1) {
printf("Error\n");
          break;
   }
   else {
         for (int i = m-1; i > 1; i--)
    {          m *=i;
    }
   }
```

Рис. 1. Пример снятия анафорики в процедурно-ориентированном языке C

Поэтому перед построением частотного словаря для программ, содержащих подобные фрагменты кода, необходимо сначала выявить анафорические отношения — отношения между языковыми выражениями, при котором в смысл одного выражения входит отсылка к другому, ранее упомянутому языковому выражению, — провести все замены, а затем уже формировать частотный словарь с учетом увеличения значений частоты встречаемости слов, входящих в анафорические отношения.

2. Построение частотного словаря и исследование программного средства Rhino

Для выявления лингвистических свойств компьютерных программ рассмотрены исходные коды, написанные на языке Java. Из них сформированы частотные словари для каждой исследуемой программы. Частотный словарь представляет из себя таблицу, в которой отражено соответствие между лексемами исследуемой программы, их частотой и рангом. Данные в таблице отсортированы по убыванию частоты (Таблица 1).

В качестве примера была исследована программа Rhino — интерпретатор языка ECMAScript с открытыми исходными кодами на языке Java, произведенный компанией Mozilla Foundation [15].

Для анализа были взяты 18 версий этого программного средства, выпущенных в период с 1999 по 2009 год. Для каждой версии был построен частотный словарь

Таблица 1. Устройство частотного словаря программы на языке Java

Ранг	Частота	Лексемы
64	1195	[int]
63	898	[java.lang.Object]
62	682	[java.lang.String]
61	568	[org.mozilla.javascript.Scriptable]
60	528	[byte]
59	495	[java.lang.StringBuffer.append()]
58	295	[org.mozilla.javascript.Node]
57	274	[double]
56	243	[boolean]
55	194	[org.mozilla.javascript.Context]
54	173	[short]
53	158	[char]
52	132	[org.mozilla.javascript.Function]
51	124	[java.lang.String.charAt()]
50	120	[java.lang.Class]
49	104	[long]
48	96	[org.mozilla.classfile.ClassFileWriter.add()]
47	94	[java.lang.String.length()]
46	70	[org.mozilla.javascript.Context.reportRuntimeError()]
45	66	[org.mozilla.javascript.Source.append(), ...LineBuffer.match()]
44	64	[org.mozilla.javascript.ScriptRuntime.toNumber()]

одинакового устройства (Таблица 1), в котором каждой лексеме была сопоставлена частота ее встречаемости, далее был построен график зависимости ранга от частоты встречаемости лексем и проведена аппроксимация этого графика по модели (4) (Рис. 2). Заметим, что количество анализируемых лексем в рамках одной версии значительно — например, для одной версии 1.7R1 количество подсчитанных лексем составило 445930 единиц. Следовательно, можно считать выборку в данном методе исследования репрезентативной.

Сравнительный анализ результатов экспериментов позволил выявить следующие закономерности:

1. Все графики линейны в области низких рангов т.е. $r = \omega, 1 \leqslant r \leqslant n$, где n — некоторое целое положительное число, для всех версий разное. Линейная область графиков соответствует разделам компьютерных программ, в которых описаны, например, функции организации интерфейса пользователя, функции организации ввода-вывода, открытие файлов с данными для чтения/записи. В области высоких рангов сосредоточены базовые модули программы и примитивные типы данных, а в средней части — только базовые модули.

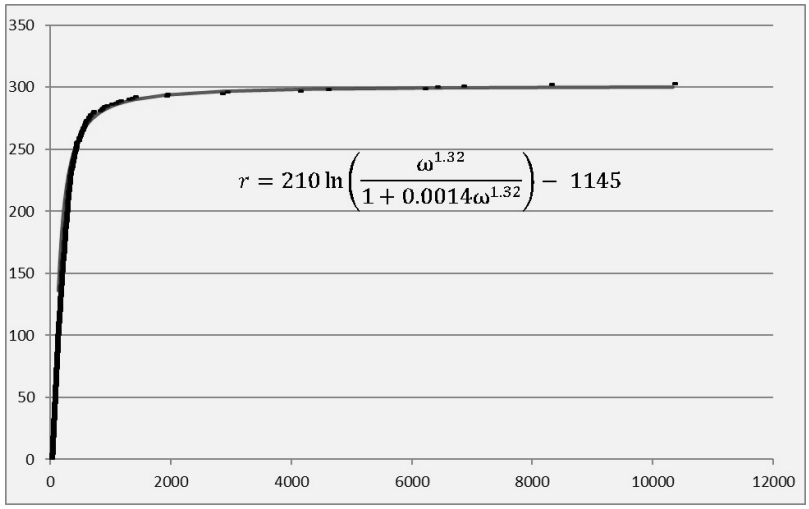

Рис. 2. График зависимости ранга от частоты и аппроксимирующая кривая для версии 1.7R1 (ось абсцисс — частота ω, ось ординат — ранг r)

2. Все графики аппроксимируются функциями вида

$$r = Aln\frac{\omega^{\gamma}}{1 + \alpha\omega^{\gamma}} - c. \tag{5}$$

Линейная часть зависимости не учитывалась при проведении аппроксимации. Результаты аппроксимации собраны в таблице (Рис. 3).

Основной параметр рангового распределения γ находится в пределах [1,28;1,43], что характеризует язык Java как стабильную, сбалансированную лингвистическую систему [8, с. 384], использующуюся для минимально достаточного эффективного кодирования. Разброс значений на линейных участках для различных версий предположительно связан с преимущественными доработками версий языка в части модулей, касающихся организации интерфейсов с пользователем и операционной системой и устройствами, так как наблюдается количественный рост длины этих участков в последующих версиях языка.

3. Заключение

Авторы выражают глубокую признательность М. А. Князевой, предложившей расширить метод ранговых распределений В. П. Маслова для анализа формальных языков, за идею исследования и определение основных параметров и правил применения метода. Полученные результаты и эта статья — дань уважения и памяти прекрасному человеку и признанному специалисту в области высокоуровневых языков программирования.

На основе анализа открытых исходных кодов программ на языке Java различных версий с учетом анафорических отношений были построены частотные словари, выполнена аппроксимация кривой рангового распределения одного вида, определены

Версия Rhino	Формула	α	γ	Длина линейного участка, ед.	Объем кода, Кб
1_4R3	$r = 35\ln\left(\frac{\omega^{1.349}}{1+0.0081\omega^{1.349}}\right) - 111$	0,0081	1,35	29	1065
1_5R1	$r = 40\ln\left(\frac{\omega^{1.4}}{1+0.00181\omega^{1.4}}\right) - 157$	0,00181	1,4	40	1631
1_5R2	$r = 65\ln\left(\frac{\omega^{1.36}}{1+0.0019\omega^{1.36}}\right) - 282$	0,0019	1,36	47	1993
1_5R3	$r = 75\ln\left(\frac{\omega^{1.32}}{1+0.002\omega^{1.32}}\right) - 316$	0,002	1,32	56	1996
1_5R4	$r = 105\ln\left(\frac{\omega^{1.3}}{1+0.002\omega^{1.3}}\right) - 478$	0,002	1,3	67	2123
1_5R4_1	$r = 105\ln\left(\frac{\omega^{1.32}}{1+0.0015\omega^{1.32}}\right) - 490$	0,0015	1,32	79	2127
1_5R5	$r = 128\ln\left(\frac{\omega^{1.28}}{1+0.0022\omega^{1.28}}\right) - 580$	0,0022	1,28	81	2214
1_6R1	$r = 170\ln\left(\frac{\omega^{1.34}}{1+0.002\omega^{1.34}}\right) - 838$	0,002	1,34	104	2460
1_6R2	$r = 190\ln\left(\frac{\omega^{1.34}}{1+0.002\omega^{1.34}}\right) - 940$	0,002	1,34	88	2460
1_6R3	$r = 200\ln\left(\frac{\omega^{1.36}}{1+0.0018\omega^{1.36}}\right) - 1008$	0,0018	1,36	81	2508
1_6R4	$r = 210\ln\left(\frac{\omega^{1.36}}{1+0.0014\omega^{1.36}}\right) - 1103$	0,0014	1,36	99	2509
1_6R5	$r = 220\ln\left(\frac{\omega^{1.34}}{1+0.0014\omega^{1.34}}\right) - 1153$	0,0014	1,34	105	2532
1_7R1	$r = 210\ln\left(\frac{\omega^{1.32}}{1+0.0014\omega^{1.32}}\right) - 1145$	0,0014	1,32	104	2866
1_7R2-RC1	$r = 220\ln\left(\frac{\omega^{1.38}}{1+0.0006*\omega^{1.38}}\right) - 1298$	0,0006	1,38	99	2979
1_7R2-RC2	$r = 220\ln\left(\frac{\omega^{1.38}}{1+0.0006\omega^{1.38}}\right) - 1280$	0,0006	1,38	120	2993
1_7R2-RC3	$r = 220\ln\left(\frac{\omega^{1.43}}{1+0.0005\omega^{1.43}}\right) - 1320$	0,0005	1,43	105	2996

Рис. 3. Таблица параметров аппроксимации графиков

значения характеристик аппроксимирующей функции для каждой версии. Выявлена общая особенность для частотных словарей компьютерных программ — наличие линейного участка графика в области низких рангов. Предположительно данная особенность характерна для всех исходных кодов программ на любом высокоуровневом языке программирования, так как соответствует лексемам, использующимся для организации взаимодействия программы и операционной системы в части чтения и записи данных и диалога с пользователем.

Описанный подход дает возможность по-новому анализировать формальные языки компьютерных программ с учетом их особенностей, использовать метод ранговых распределений так же результативно, как это давно применяется в исследовании естественных литературных языков, что открывает новые перспективы при решении задач поиска неочевидных закономерностей в технологии разработки программных средств.

Список литературы

[1] М. В. Арапов, Е. Н. Ефимова, Ю. А. Шрейдер, “О смысле ранговых распределений”, *Научно-техническая информация*, **2**:1, (1975).

[2] М. А. Гузев, Е. Ю. Никитина, “Ранговый анализ Уголовного Кодекса РФ (на примере экономических преступлений)”, *Дальневост. матем. журн.*, **10**:2, (2010).

[3] М. А. Гузев, Н. Н. Крадин, Е. Ю. Никитина, “Ранговый анализ жизненного цикла политий”, *Дальневост. матем. журн.*, **17**:2, (2017).

[4] В. А. Столбов, М. Д. Шарыгин, *Введение в экономическую и социальную географию. Учебное пособие для вузов*, Дрофа, М., 2007.

[5] Richard Koch, *The 80/20 Principle*, Nicholas Brealey Publishing, London, 1997.

[6] Н. Н. Чурсин, *Популярная информатика*, Техника, К., 1982.

[7] B. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, New York, 1977.

[8] В. И. Гнатюк, “Закон оптимального построения техноценозов”, *Ценологические исследования*, **29**, (2005).

[9] V. P. Maslov, “Quantum Linguistic Statistics”, *Russian Journal of Mathematical Physics*, **13**:3, (2006).

[10] В. П. Маслов, Т. В. Маслова, “О законе Ципфа и ранговых распределениях в лингвистике и семиотике”, *Мат. Заметки*, **80**:5, (2006).

[11] В. П. Маслов, “Закон «отсутствия предпочтения» и соответствующие распределения в частотной теории вероятностей”, *Мат. Заметки*, **80**:2, (2006).

[12] В. П. Маслов, “Фазовые переходы нулевого рода и квантование закона Ципфа”, *Теоретическая и математическая физика*, **150**:1, (2007).

[13] В. П. Маслов, *Квантовая экономика*, Наука, М., 2006.

[14] M. A. Guzev, N .N. Kradin, E. Y. Nikitina, “The Imperial Curve of Large Polities”, *Social Evolution & History*, **16**:2, (2017).

[15] Mozilla Foundation, “MDN web docs. Rhino”, 2020, http://www.mozilla.org/rhino/.

Поступила в редакцию
27 октября 2020 г.

Guzev M. A., *Knyazeva M. A.*, *Moskalev I. I., Nikitina E. Y.* Rank analysis of computer programs. *Far Eastern Mathematical Journal.* 2020. V. 20. No 2. P. 155–163.

ABSTRACT

This paper describes the application of the rank analysis method for formal languages on the example of the source codes of programs in the Java language. When describing the technology for compiling frequency dictionaries, attention is focused on how the presence of anaphoric content in computer programs is taken into consideration. When discussing the results, it was noted that the structural components of programs for various purposes correspond to certain areas of the graph of the dependence of rank on the frequency of occurrence of a lexeme.

Key words: *rank distributions, Zipf's Low, formal languages, frequency dictionaries.*