Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №2»**

“Элементы теории информации. Параметры и характеристики дискретных информационных систем”

**Выполнил:** студент 3 курса

4 группы специальности ПОИТ

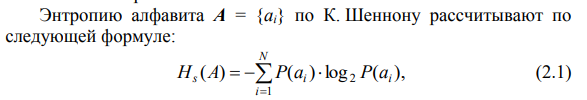
Мозолевский Александр Дмитриевич

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2023

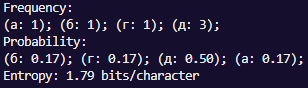
**Рассчитать энтропию указанного преподавателем алфавитов: один – на латинице, другой – на кириллице(по формуле (2.1) – перейти от частоты появления каждого символа алфавита к соответствующей вероятности); в качестве входного может быть принят произвольный электронный текстовый документ на основе соответствующего алфавита; частоты появления символов алфавитов оформить в виде гистограмм (можно воспользоваться приложением MS Excel):**

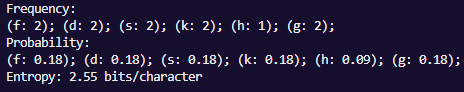


Реализация на языке GO:

|  |
| --- |
| func calculateEntropy(file \*os.File) float64 {  freq := make(map[rune]int)  scanner := bufio.NewScanner(bufio.NewReader(file))  scanner.Split(bufio.ScanRunes)  for scanner.Scan() {  line := scanner.Text()  for len(line) > 0 {  r, size := utf8.DecodeRuneInString(line)  line = line[size:]  if unicode.IsLetter(r) {  freq[unicode.ToLower(r)]++  }  }  }  fmt.Println("Frequency: ")  for k, v := range freq {  fmt.Print("(", string(k), ": ", v, "); ")  }  fmt.Println()  err := createReport(freq, "Entropy")  if err != nil {  fmt.Println("Error writing report:", err)  }  total := findTotal(freq)  fmt.Println("Probability: ")  prob := make(map[rune]float64)  for symbol, count := range freq {  prob[symbol] = float64(count) / float64(total)  fmt.Printf("(%c: %0.2f); ", symbol, prob[symbol])  }  fmt.Println()  return shennonFano(prob)  } |

Полученная энтропия:



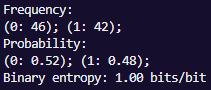


**Для входных документов, представленных в бинарных кодах, определить энтропию бинарного алфавита.**

Реализация на языке GO:

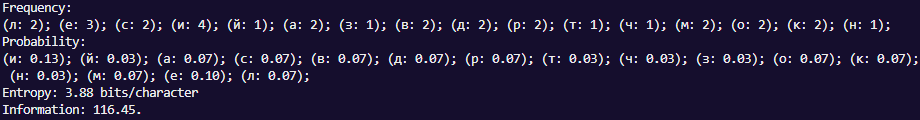
|  |
| --- |
| func calculateEntropyBinary(file \*os.File) float64 {  freq := make(map[byte]int)  buffer := make([]byte, 1024)  n, err := file.Read(buffer)  if err != nil && err != io.EOF {  panic(err)  }  for i := 0; i < n; i++ {  for j := 7; j >= 0; j-- {  b := (buffer[i] >> uint(j)) & 1  freq[byte(b)]++  }  }  fmt.Println("Frequency: ")  for k, v := range freq {  fmt.Print("(", k, ": ", v, "); ")  }  fmt.Println()  total := findTotal(freq)  fmt.Println("Probability: ")  prob := make(map[byte]float64)  for bit, count := range freq {  prob[bit] = float64(count) / float64(total)  fmt.Printf("(%d: %0.2f); ", bit, prob[bit])  }  fmt.Println()  return shennonFano(prob)  } |

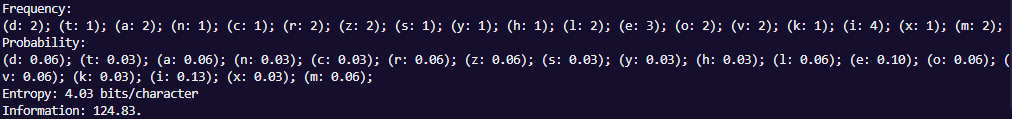
Полученная энтропия:



**Используя значения энтропии алфавитов, полученных в пп. а) и б), подсчитать количество информации в сообщении, состоящем из собственных фамилии, имени по отчества (на основе исходного алфавита – а) и в кодах ASCII – б); объяснить полученный результат;**

Полученная информация:





**Выполнить задание п. в) при условии, что вероятность ошибочной передачи единичного бита сообщения составляет: 0.1; 0.5; 1.0.**

Полученный результат:

