



**«Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)»**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

О Т Ч Е Т

по лабораторной работе № 1

Дисциплина: Теория информации

Название лабораторной работы:

Студент гр. ИУ6-54Б

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

(И.О. Фамилия)

Москва, 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Расчёт ngd для пар наиболее часто встречающихся слов в тексте.....	4
2 Построение марковских цепей.....	
2.1 Построение марковской цепи в 0 приближении.....	
2.2 Построение марковских цепей в 1 и 2 приближении.....	
ЗАКЛЮЧЕНИЕ.....	
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	

ВВЕДЕНИЕ

Целью лабораторной работы является

Задачами лабораторной работы являются расчёт ngd 30-ти пар наиболее часто встречающихся в тексте слов и построение на основании этого же текста марковских цепей в 0, 1 и 2 приближении.

1 Расчёт *ngd* для пар наиболее часто встречающихся слов в тексте

В качестве текста для анализа был выбран текст труда Карла Маркса «Капитал» [1]. Текст достаточно объёмный, поэтому не будет включён в текст отчёта.

Алгоритм поиска наиболее часто встречающихся слов представлен в классе CountMaker, показаном на листинге 1.

Листинг 1 — Код класса CountMaker 1

```
namespace TI_Lab1.MainBlock
{
    internal class CountMaker
    {
        public CountMaker()
        {
            TxtIntroduction txtIntroduction = new TxtIntroduction();
            ExelOutputService exelOutputService = new ExelOutputService();

            var text = txtIntroduction.InputFromSource("forinput.txt").Result;
            var fulltext = txtIntroduction.InputFromSource("kapital.txt").Result;

            var tmp = new WordTopMaker();

            var words = tmp.MakeWordsList(text).Result.ToList();
            var fullWords = tmp.MakeWordsList(fulltext).Result.ToList();

            var textWithTop = new WordsWithCounter(words);
            var fullTextWithTop = new WordsWithCounter(fullWords);

            WordTopMaker wordTopMaker = new WordTopMaker(textWithTop);
            WordTopMaker wordTopMaker2 = new WordTopMaker(fullTextWithTop);

            var topFromInput = wordTopMaker.MakeTop(words).Result;
            var topFromFull = wordTopMaker2.MakeTop(fullWords).Result;

            var cgi = new CounterOfGoogleIndex(topFromInput, topFromFull);

            var tmb = cgi.CountGoogleIndex(topFromInput.Word).Result;
            var tmb2 = cgi.CountGoogleIndex(topFromFull.Word).Result;

            exelOutputService.ExportToExcel(tmb, "TopWordsFromInput.xlsx");
            exelOutputService.ExportToExcel(tmb2, "TopWordsFromFull.xlsx", true);
        }
    }
}
```

TxtIntroduction используется для считывания текста и подсчета количества букв в тексте показан на листинге 2.

Листинг 2 — Код класса TxtIntroduction

```
namespace TI_Lab1.Processing
{
    internal class TxtIntroduction
    {
        private string filePath;
        private readonly Alphabet alphabet;

        public string FilePath { get => filePath; set => filePath = value; }

        public async Task<string> InputFromSource(string source)
        {
            try
            {
                FilePath = source;
                if (!File.Exists(filePath))
                {
                    Exception ex = new Exception("Файл не найден.");
                }

                string content = File.ReadAllText(filePath).ToLower();
                return content;
            }
            catch
            {
                Exception ex = new Exception("Ошибка при чтении файла.");
                return ex.Message;
            }
        }

        public async Task<LetterCounter> CountTheLetters(string content,
LetterCounter letterCounter)
        {
            foreach (char c in content)
            {
                if (letterCounter.KeyValuePairs.ContainsKey(c))
                {
                    letterCounter.KeyValuePairs[c]++;
                }
            }

            for (int i = 0; i < content.Length - 1; i++)
            {
                var firstChar = content[i];
                var secondChar = content[i + 1];

                if (letterCounter.KeyValuePairs.ContainsKey(firstChar))
                    letterCounter.KeyValuePairs[firstChar]++;

                if (letterCounter.pairCount.ContainsKey(firstChar) &&
letterCounter.pairCount[firstChar].ContainsKey(secondChar))
                {
                    letterCounter.pairCount[firstChar][secondChar]++;
                }
            }
            return letterCounter;
        }
    }
}
```

WordTopMaker – модуль, который подсчитывает слова. Он показан на листинге 3.

Листинг 3 — Код класса WordTopMaker

```
namespace TI_Lab1.Processing
{
    internal class WordTopMaker
    {
        private WordsWithCounter words;

        public WordTopMaker()
        {
        }

        public WordTopMaker(WordsWithCounter wwc)
        {
            words = wwc;
        }

        public async Task<WordsWithCounter> MakeTop(List<string> allWords)
        {
            foreach (var word in allWords)
            {
                if (words.Word[word] != null)
                    words.Word[word]++;
                else
                    words.Word[word] = 0;
            }
            return words;
        }

        public async Task<IEnumerable<string>> MakeWordsList(string text)
        {
            return text.Split(new char[] { ' ', ',', '.', '!', '?', '"', '\'', '[',
                ']', ':', ';', '(', ')', '-' },
                StringSplitOptions.RemoveEmptyEntries);
        }
    }
}
```

Для хранения топа используется класс WordsWithCunter, который показан на листинге 4.

Листинг 4 — Код класса WordsWithCunter

```
private TxtIntroduction txtIntroduction = new TxtIntroduction();

public Dictionary<string, int> Word = new Dictionary<string, int>();

public WordsWithCounter(List<string> fulltext)
{
    foreach (var word in fulltext)
    {
        Word[word] = 0;
    }
}
```

Для сортировки топа используется модуль CounterOfGoogleIndex, который показан на листинге 5.

Листинг 5 — Код класса CounterOfGoogleIndex

```
namespace TI_Lab1.Processing
{
    internal class CounterOfGoogleIndex
    {
        private WordsWithCounter topFromText;
        private WordsWithCounter topFromFull;
        private List<string> topWords;
        private List<(string, string)> topWordsPairs;
        public CounterOfGoogleIndex(WordsWithCounter small, WordsWithCounter big)
        {
            topFromText = small;
            topFromFull = big;
            topWords = new List<string>();
            topWordsPairs = new List<(string, string)>();
        }

        public async Task<Dictionary<string, int>>
        CountGoogleIndex(Dictionary<string, int> vls)
        {
            var top = vls.OrderByDescending(x => x.Value);
            int cnt = 0;
            var topWords = new Dictionary<string, int>();
            foreach (var word in top)
            {
                topWords[word.Key] = word.Value;
            }
            return topWords;
        }
    }
}
```

Результатом выполнения после удаления союзов, предлогов, местоимений и частиц является 6 слов:

- 1) производство — 7 вхождение;
- 2) более — 6 вхождений;
- 3) стоимость — 6 вхождений;
- 4) исследования — 4 вхождений;
- 5) развитие — 4 вхождений;
- 6) форма — 4 вхождений.

Для интереса, посмотрим на 3 самых популярных слова в всем тексте.

- 1) стоимость — 5318 вхождение;
- 2) капитал — 4753 вхождений;
- 3) производство — 2454 вхождений.

Результаты вычисления *ngd* для всех пар из них представлены в таблице 1.

Таблица 1 — *Ngd* пар слов из текста

Слово 1	Слово 2	<i>ngd</i>
производство	более	0,6460920355466139
производство	стоимость	-4,357744335075464
производство	исследования	-0,8053898992701236
производство	развитие	-6,81317621922346
производство	форма	2,954967341273358
более	производство	0,15597133001553115
более	стоимость	0,6599964636813697
более	исследования	-0,9260615377821461
более	развитие	3,1626131320215047
более	форма	-4,186536694692931
стоимость	производство	-4,8228758071120765
стоимость	более	0,73578469314351
стоимость	исследования	-4,30857379544644
стоимость	развитие	-0,3068655711456081
стоимость	форма	5,662735702595762
исследования	производство	-0,8935974745420583
исследования	более	-0,9504813990857083
исследования	стоимость	-4,3206870232042665
исследования	развитие	-6,316599255373193
исследования	форма	-2,9089064989372986
развитие	производство	-6,802364248760696
развитие	более	11,086724706382784
развитие	стоимость	-0,19541822762957853
развитие	исследования	9,721181035753467
развитие	форма	9,721181035753467
форма	производство	4,721426030164025
форма	более	-5,136463665983541
форма	стоимость	4,510937894529924
форма	исследования	-4,340161208686698

форма

развитие

2,986117212958122

2 Построение марковских цепей

2.1 Построение марковской цепи в 0 приближении

Код для построения представлен на листинге 2.

Листинг 2 — Построение 0 приближения

```
public async Task<string> MakeZeroString()
{
    return MakeString(alphabet.RusAlphabet).Result;
}
private async Task<Dictionary<char, string>> CreateRandomToMatrix()
{
    string element;
    var result = new Dictionary<char, string>();

    foreach (var letters in letterCounter.pairCount)
    {
        result[letters.Key] = CreateRandomToList(letters.Value).Result;
    }

    return result;
}
private async Task<string> MakeString(string alpha)
{
    var res = "";
    for (var i = 0; i < count; i++)
    {
        var index = random.Next(alpha.Length);
        res += alpha[index];
    }
    if (res.Count() == 0)
        return new Exception("Предложение не создано").Message;

    return res.ToString();
}
```

Результат работы представлен на рисунке 1.

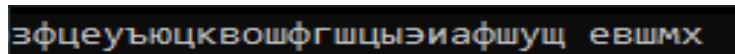


Рисунок 1 — Цепь в 0 приближении

2.2 Построение марковских цепей в 1 и 2 приближении

Код программы вычисления количества символов и пар символов и построения цепей в 1 и 2 приближении представлен на листинге 3.

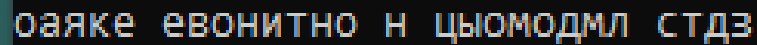
Листинг 3 — Построение 1 и 2 приближений

```
public async Task<LetterCounter> CountTheLetters(string content, LetterCounter letterCounter)
{
    foreach (char c in content)
    {
        if (letterCounter.KeyValuePairs.ContainsKey(c))
        {
            letterCounter.KeyValuePairs[c]++;
        }
    }
}
```

Продолжение листинга 3

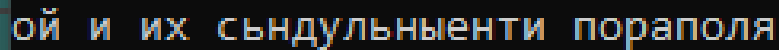
```
    }  
    }  
  
    for (int i = 0; i < content.Length - 1; i++)  
    {  
        var firstChar = content[i];  
        var secondChar = content[i + 1];  
  
        if (letterCounter.KeyValuePairs.ContainsKey(firstChar))  
            letterCounter.KeyValuePairs[firstChar]++;  
  
        if (letterCounter.pairCount.ContainsKey(firstChar) &&  
letterCounter.pairCount[firstChar].ContainsKey(secondChar))  
        {  
            letterCounter.pairCount[firstChar][secondChar]++;  
        }  
    }  
    return letterCounter;  
}
```

Результаты работы представлены на рисунках 2-3.



оаяке евонитно н цыомодмл stdз

Рисунок 2 — Цепь в 1 приближении



ой и их сындульныенти пораполя

Рисунок 3 — Цепь во 2 приближении

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Карл Маркс. Капитал [Электронный ресурс]. - URL:
https://royallib.com/book/marks_karl/kapital.html?ysclid=m2t4fdnva3823043201
(дата обращения 02.10.2024)