

Министерство образования Республики Беларусь
Учреждение образования Белорусский государственный университет
информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №4
“ Работа с пакетом SPARK ”

Проверил:

Выполнил:

Минск 2023

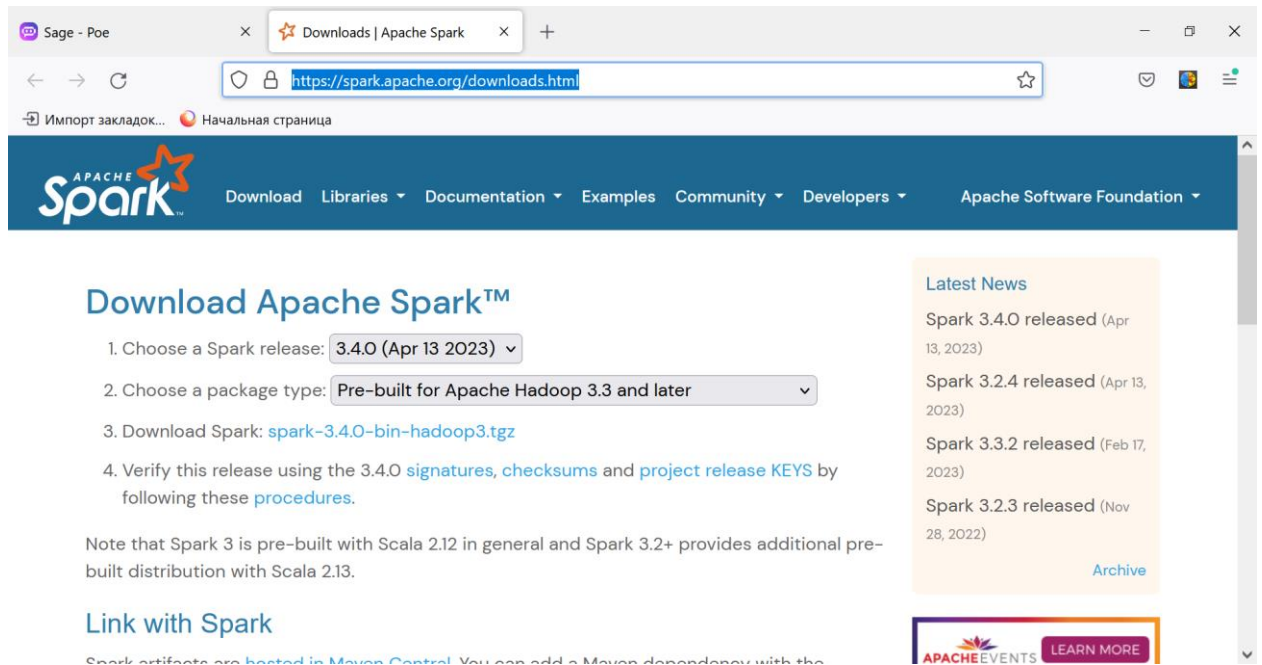
1. Цель

Изучить технику обработки текста в SPARK Scala.

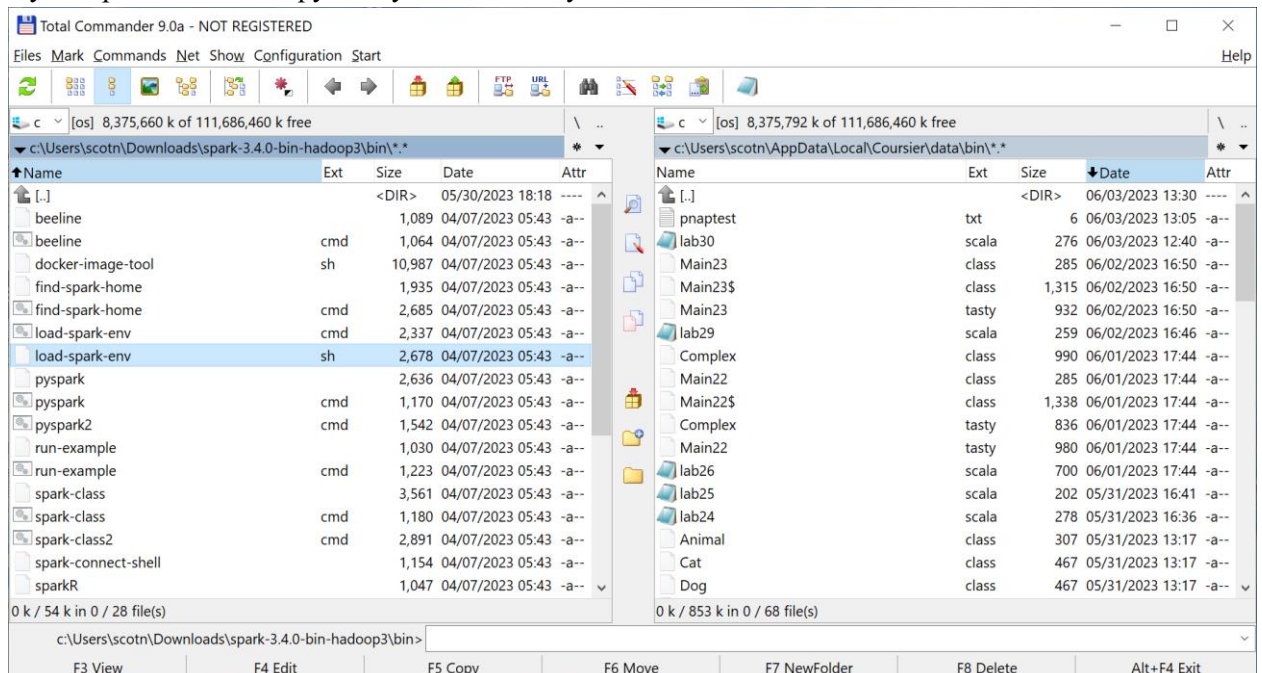
2. Краткие теоретические сведения

Загрузим SPARK с сайта <https://spark.apache.org/downloads.html>

Вот его окно



Нужно распаковать загруженную библиотеку. На моей машине это



Чтобы запустить SPARK, нужно открыть командное окно в режиме администратора и ввести команду

conf: org.apache.spark.SparkConf = [org.apache.spark.SparkConf@1e2a39b7](#)

Вот теперь мы читаем текстовый файл:

```
scala> val lines = spark.read.textFile("c:\\Users\\scotn\\Downloads\\pnaptext.txt")
```

Прописываем полный путь. Да еще дублируем символ-разделитель - \\

Строки прочтены

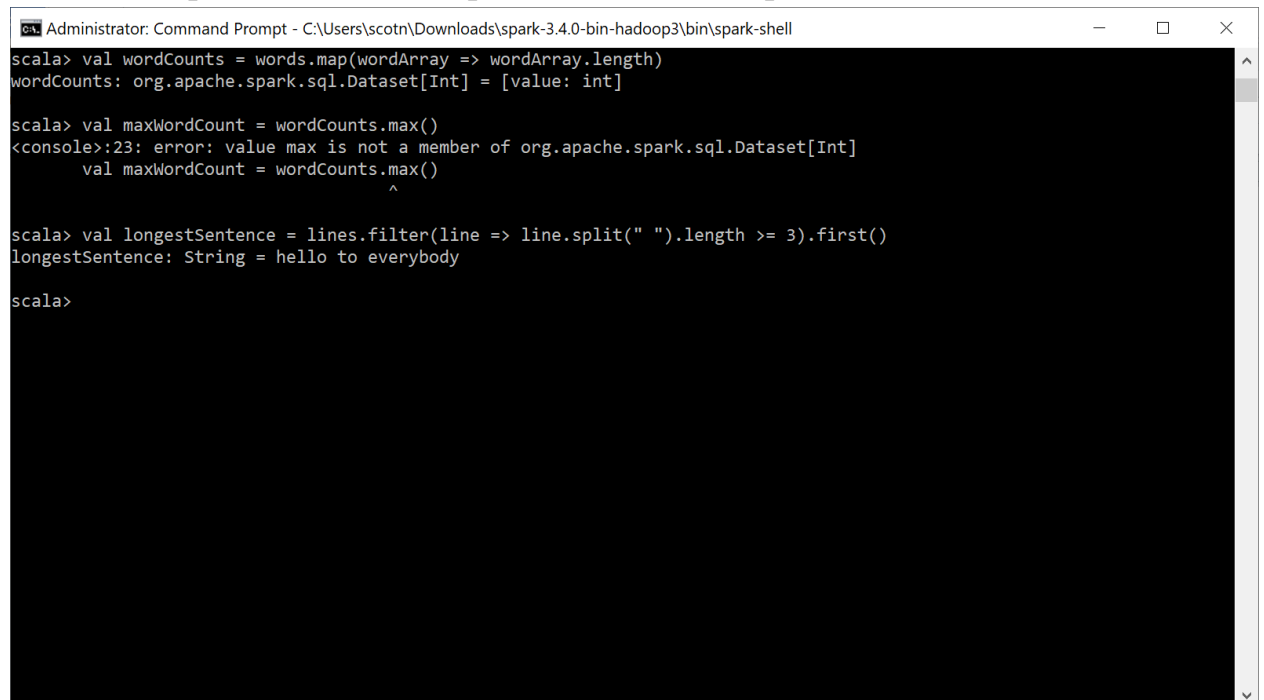
```
scala> val lines = spark.read.textFile("c:\\Users\\scotn\\Downloads\\pnaptext.txt")
lines: org.apache.spark.sql.Dataset[String] = [value: string]
```

Получаем массив всех строк

```
scala> val words = lines.map(line => line.split(" "))
```

```
words: org.apache.spark.sql.Dataset[Array[String]] = [value: array<string>]
```

Выводим предложение, содержащее не менее трех слов



```
Administrator: Command Prompt - C:\Users\scotn\Downloads\spark-3.4.0-bin-hadoop3\bin\spark-shell
scala> val wordCounts = words.map(wordArray => wordArray.length)
wordCounts: org.apache.spark.sql.Dataset[Int] = [value: int]

scala> val maxWordCount = wordCounts.max()
<console>:23: error: value max is not a member of org.apache.spark.sql.Dataset[Int]
      val maxWordCount = wordCounts.max()
                                ^

scala> val longestSentence = lines.filter(line => line.split(" ").length >= 3).first()
longestSentence: String = hello to everybody

scala>
```

В нашем текстовом файле хранится такой текст

hello to everybody

this is first attempt in Spark text processing

Be careful with people No all of them sympathize you

Выводим строки из текстового файла

```
val longestSentence2 = lines.foreach(line => println(line))
```

```
Administrator: Command Prompt - C:\Users\scotn\Downloads\spark-3.4.0-bin-hadoop3\bin\spark-shell
scala> val longestSentence2 = lines.foreach(line => println(line))
hello to everybody
this is first attemp in Spark text processing
longestSentence2: Unit = ()

scala> _
```

Повторение.

Начинаем с чтения текстового файла

```
scala> val lines = spark.read.textFile("c:\\Users\\scotn\\Downloads\\pnapttext.txt")
lines: org.apache.spark.sql.Dataset[String] = [value: string]
```

Расщепляем текст на отдельные слова. При этом используем регулярное выражение `\\W+` (означает любую последовательность символов)

```
scala> val wordsRDD = lines.flatMap(line => line.split("\\W+"))
wordsRDD: org.apache.spark.sql.Dataset[String] = [value: string]
```

Слова приводим к нижнему регистру

```
scala> val lowercaseWordsRDD = wordsRDD.map(word => word.toLowerCase())
lowercaseWordsRDD: org.apache.spark.sql.Dataset[String] = [value: string]
```

Создаем множество stop-слов

```
scala> val stopWords = Set("a", "an", "the", "and", "but", "or", "for", "of")
stopWords: scala.collection.immutable.Set[String] = Set(for, but, a, or, an, of, and, the)
```

Удаляем стоп-слова из текста

```
scala> val filteredWordsRDD = lowercaseWordsRDD.filter(word => !stopWords.contains(word))
filteredWordsRDD: org.apache.spark.sql.Dataset[String] = [value: string]
```

Выводим отфильтрованное множество слов

```
Command Prompt - C:\Users\scotn\Downloads\spark-3.4.0-bin-hadoop3\bin\spark-shell

scala> filteredWordsRDD.collect().foreach(println)
hello
to
everybody
this
is
first
attemp
in
spark
text
processing
scala> _
```

Отфильтруем слова, начинающиеся на s:

```
scala> val filteredWordsRDD2 = wordsRDD.filter(word => word.startsWith("s"))
filteredWordsRDD2: org.apache.spark.sql.Dataset[String] = [value: string]
```

Выведем их

```
scala> val filteredWordsRDD2 = filteredWordsRDD.filter(word => word.startsWith("s"))
filteredWordsRDD2: org.apache.spark.sql.Dataset[String] = [value: string]

scala> filteredWordsRDD2.collect().foreach(println)
spark
```

Итак, мы научились формировать множества слов и фильтровать их.

Познакомимся чуть ближе с регулярными выражениями

Создадим паттерн

```
scala> val pattern = "^s.*r*k$"
pattern: String = ^s.*r*k$
```

Этот паттерн определяет все слова, начинающиеся на s, содержащие где-то посередине r и заканчивающиеся на k

Сформируем отфильтрованное множество по паттерну

```
scala> val filteredWordsRDD3 = filteredWordsRDD.filter(word => word.matches(pattern))
filteredWordsRDD3: org.apache.spark.sql.Dataset[String] = [value: string]
```

Выведем результат на консоль

```
scala> filteredWordsRDD3.collect().foreach(println)
spark
```

3. Индивидуальное задание

Вариант 1.

Создать собственный текстовый файл на английском или немецком языке – 4-5 предложений.

1. Вывести все слова из текстового файла, исключая stop-слова;
2. Вывести все слова, содержащие букву t;
3. Вывести все слова, заканчивающиеся на ing;
4. Вывести все слова, вторая буква которых a;
5. Вывести все слова, последняя буква которых s;
6. Вывести каждое второе слово.

Листинг программы:

Код программы:

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

val text = sc.textFile("D:\\SCALA_TEXT.txt")

val stopWords = Seq("the", "and", "in", "of", "to", "a", "for") // Пример списка stop-слов
val filteredWords = text.flatMap(line => line.split("\\s")).filter(word
=> !stopWords.contains(word))
println("1. Слова, исключая stop-слова:" + stopWords)
filteredWords.collect().foreach(println)

val wordsWithT = text.flatMap(line => line.split("\\s")).filter(word => word.contains("t"))
println("2. Слова, содержащие букву 't':")
wordsWithT.collect().foreach(println)

val wordsEndingWithIng = text.flatMap(line => line.split("\\s")).filter(word =>
word.endsWith("ing"))
println("3. Слова, заканчивающиеся на 'ing':")
wordsEndingWithIng.collect().foreach(println)

val wordsWithSecondLetterA = text.flatMap(line => line.split("\\s")).filter(word => word.length
> 1 && word(1) == 'a')
println("4. Слова, вторая буква которых 'a':")
wordsWithSecondLetterA.collect().foreach(println)

val wordsWithLastLetterS = text.flatMap(line => line.split("\\s")).filter(word =>
word.endsWith("s"))
println("5. Слова, последняя буква которых 's':")
wordsWithLastLetterS.collect().foreach(println)
```

```

val everySecondWord = text.flatMap(line => line.split("\\s")).zipWithIndex.filter { case (_,
index) => index % 2 == 1 }.map(_._1)
println("6. Каждое второе слово:")
everySecondWord.collect().foreach(println)

sc.stop()

```

1. Вывести все слова из текстового файла, исключая stop-слова

```

val stopWords = Seq("the", "and", "in", "of", "to", "a", "for") // Пример списка stop-
слов
val filteredWords = text.flatMap(line => line.split("\\s")).filter(word
=> !stopWords.contains(word))
println("1. Слова, исключая stop-слова:" + stopWords)
filteredWords.collect().foreach(println)

```

Работа функции:

```

scala> println("1. Слова, исключая stop-слова:" + stopWords)
1. Слова, исключая stop-слова:List(the, and, in, of, to, a, for)

scala> filteredWords.collect().foreach(println)
I
have
always
had
deep
passion
programming.
The
thrill
writing
code
seeing
it
come
life
excites
me
every
day.
It's
world
where
creativity
knows
no
bounds,
problem-solving
is
way
life.
Whether
it's
building
software
or
developing
algorithms,
I
find
immense
satisfaction
art
programming.

```

2. Вывести все слова, содержащие букву t

```

val wordsWithT = text.flatMap(line => line.split("\\s")).filter(word =>
word.contains("t"))
println("2. Слова, содержащие букву 't':")
wordsWithT.collect().foreach(println)

```


Работа функции:

```
scala> println("2. Слова, содержащие букву 't':")
2. Слова, содержащие букву 't':

scala> wordsWithT.collect().foreach(println)_
thrill
writing
it
to
excites
It's
creativity
Whether
it's
software
algorithms,
satisfaction
the
art
```

3. Вывести все слова, заканчивающиеся на ing

```
val wordsEndingWithIng = text.flatMap(line => line.split("\s")).filter(word =>
word.endsWith("ing"))
println("3. Слова, заканчивающиеся на 'ing':")
wordsEndingWithIng.collect().foreach(println)
```

Работа функции:

```
scala> println("3. Слова, заканчивающиеся на 'ing':")
3. Слова, заканчивающиеся на 'ing':

scala> _ wordsEndingWithIng.collect().foreach(println)
writing
seeing
problem-solving
building
developing
```

4. Вывести все слова вторая буква которых a

```
val wordsWithSecondLetterA = text.flatMap(line => line.split("\s")).filter(word =>
word.length > 1 && word(1) == 'a')
println("4. Слова, вторая буква которых 'a':")
wordsWithSecondLetterA.collect().foreach(println)
```

Работа функции:

```
scala> _ println("4. Слова, вторая буква которых 'a':")
4. Слова, вторая буква которых 'a':

scala> wordsWithSecondLetterA.collect().foreach(println)
have
had
passion
day.
way
satisfaction
```

5. Вывести все слова, последняя буква которых s.

```
val wordsWithLastLetterS = text.flatMap(line => line.split("\\s")).filter(word =>
word.endsWith("s"))
println("5. Слова, последняя буква которых 's':")
wordsWithLastLetterS.collect().foreach(println)
```

Работа функции:

```
scala> println("5. Слова, последняя буква которых 's':")
5. Слова, последняя буква которых 's':

scala> wordsWithLastLetterS.collect().foreach(println)
always
excites
It's
knows
is
it's
```

6. Вывести каждое второе слово

```
val everySecondWord = text.flatMap(line => line.split("\\s")).zipWithIndex.filter { case (_,
index) => index % 2 == 1 }.map(_._1)
println("6. Каждое второе слово:")
everySecondWord.collect().foreach(println)
```

Работа функции:

```
scala> println("6. Каждое второе слово:")
6. Каждое второе слово:

scala> everySecondWord.collect().foreach(println)
have
had
deep
for
The
of
code
seeing
come
life
me
day.
a
where
knows
bounds,
problem-solving
a
of
whether
building
or
algorithms,
find
satisfaction
the
of
```

Вывод

Познакомились с техникой обработки текста в SPARK Scala.