ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ АСТРАХАНСКОЙ ОБЛАСТИ
ВЫСШЕГО ОБРАЗОВАНИЯ
«АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ
АРХИТЕКТУРНО-
СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра «САПРиМ»

ТЕХНОЛОГИЯ JAVA

Методические указания по выполнению лабораторных работ
для студентов направления подготовки
09.04.02 «Информационные системы и технологии»
Программа «Искусственный интеллект в проектировании и производстве»
Направленность (профиль)
«Искусственный интеллект в проектировании городской среды»
очной и заочной форм обучения

Астрахань 2021

Составитель: к.т.н., доцент _____ /О.И. Евдошенко
(занимаемая должность, учёное степень и учёное звание) И.О.Ф.

Рецензент: д.т.н., профессор _____ /Т.В. Хошенко
(занимаемая должность, учёная степень и учёное звание) (подпись) И. О. Ф.

Методические указания к выполнению практических работ для студентов направление подготовки 09.04.02 «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды» очной и заочной форм обучения рассмотрены и одобрены на заседании кафедры «Систем автоматизированного проектирования и моделирования» ГАОУ АО ВО «АГАСУ»

Протокол №___ от_____2021

Зав.кафедрой /Евдошенко О.И.

Согласовано с УМУ ГАОУ АО ВО «АГАСУ»
_____2021

Специалист УМУ _____ / _____
подпись И.О.Ф.

Методические указания к выполнению практических работ для студентов направление подготовки 09.04.02 «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды» утверждены и рекомендованы к публикации на заседании МКН подготовки «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды».

Протокол № _ от «_____»

Председатель МКН «Информационные системы и технологии» направленность (профиль) «Информационные системы и технологии в строительстве и архитектуре»

Зав.кафедрой,
доцент, к.т.н. /О.И.Евдошенко

Оглавление

Введение

Лабораторный практикум предназначен для практической поддержки курса «Технология Java» для студентов направления подготовки 09.04.02 «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды».

Основная цель лабораторного практикума: формирование у студентов системы знаний и практических навыков в области современной разработки приложений с использованием технологий программирования на базе языка программирования Java.

Учебно-методическое пособие представляет собой сборник лабораторных работ, выполнение которых должно обеспечить формирование устойчивых умений и навыков:

- реализации алгоритма решения задачи с использованием языка программирования Java;

- анализа технической литературы;

- разработки отдельных компонентов системы и возможности интеграции их в одну систему;

- разработки безопасных консольных, веб-приложений и приложений с графическим интерфейсом, используя различные возможности языка программирования Java.

Ко всем программным средствам, рассматриваемым в лабораторных работах, приведены краткие инструкции, которые могут служить справочным материалом для самостоятельной работы. Этой же цели служит список литературы, приведенный в практикуме.

РАЗДЕЛ 1. ВВЕДЕНИЕ В JAVA

Лабораторная работа №1. Массивы, операторы (1 ч)

Задание 1

Сложиться до 10

Пользователь вводит целые числа, ноль — сигнал остановки. Гарантируется, что в какой-то момент сумма введённых чисел окажется равной 10. Программа должна вывести количество чисел, которое было введено к тому моменту, когда это случилось (в первый раз).

Формат входных данных

Несколько целых чисел, по одному в каждой строке

Формат выходных данных

Целое число — количество введенных чисел в момент, когда их сумма впервые стала равна 10

Примеры:

Входные данные

2

9

-2

1

6

6

6

0

Результат работы

4

Задание 2

Строки до 0

Пользователь вводит строки одну за другой до тех пор, пока не введёт "0". Программа должна выводить введенные строки, пока не встретилась строка содержащая "0".

Формат входных данных

Строки

Формат выходных данных

Введенные строки

Примеры:

Входные данные

Едет

по

0

Результат работы

Едет

По

Задание 3

Подсчет данных с консоли

Пользователь вводит числа в консоль по одному числу в каждой строке, ноль — сигнал остановки. Необходимо организовать считывание данных с консоли. После ввода нуля, показать на экран количество введенных чисел, их общую сумму и среднее арифметическое в одной строке.

Формат входных данных

Несколько чисел, по одному в каждой строке

Формат выходных данных

Целое число — количество введенных чисел, сумма введенных чисел - вещественное число, среднее арифметическое введенных чисел - вещественное число. Все значения выводятся через пробел.

Примеры:

Входные данные

50

10

-30

0

Результат работы

3 30.0 10.0

P.S.

Необходимо провести округление до сотых, это можно сделать, например, с помощью метода floor, который округляет аргумент до ближайшего меньшего. Если вы будете использовать другой способ, то необходимо чтобы разделителем дробной части была точка (.)

Задание 4

Елочка

С консоли вводится целое число - высота елочки в строках, с помощью символа "*" необходимо вывести в консоль елочку заданной высоты.

Формат входных данных

Целое число

Формат выходных данных

Строки

Примеры:

Входные данные

3

Результат работы

```
  *
 ***
*****
```

Задание 5

В магазине акция: скидка 5% на товары, цена которых равна или превышает 1000 рублей. Напишите программу, имитирующую работу кассового аппарата: вводятся цены покупаемых товаров, нужно вывести сумму покупки без скидки, сумму покупки с учётом скидки и размер скидки в одной строке, разделитель - пробел. Каждая цена товара вводится в новой строке. Отрицательное число - сигнал остановки ввода.

Формат входных данных

Несколько вещественных чисел в отдельной строке

Формат выходных данных

Строка содержащая: сумму покупки без скидки, сумму покупки с учётом скидки, размер скидки, разделенных пробелом

Примеры:

Входные данные

25

2000

370.35

-1

Результат работы

2395.35 2295.35 100.0

Задание 6

Перестановка в одномерном массиве

Программа получает на вход строку, содержащую три целых числа n, a и b, разделенных пробелом. n - задает размер одномерного массива, a и b указывают какие элементы поменять местами. Далее построчно вводятся n-элементов массива - целые числа. Программа должна переставить местами элементы с порядковым номером a и b (отсчет от 0) и вывести построчно

все элементы получившегося массива. Если перестановка не возможна - вывести "NO"

Формат входных данных

Строка, содержащая целые положительные числа n, a и b, разделенные пробелом n элементов массива - целые числа

Формат выходных данных

n элементов массива или "NO"

Примеры:

Входные данные

5 1 3

1

2

3

4

5

Результат работы

1

4

3

2

5

Задание 7

Подсчет отрицательных элементов в массиве

Вводятся строка, содержащая n-элементов массива - целые числа, разделенные пробелом. Программа выводит количество отрицательных элементов массива и их сумму, разделенные пробелом.

Формат входных данных

Строка, содержащая n-элементов массива - целые числа

Формат выходных данных

Количество отрицательных элементов массива и сумму отрицательных элементов в одной строке, разделенные пробелом.

Примеры:

Входные данные

1 3 -1 -10 0

Результат работы

2 -11

Задание 8

Переворот массива на 90 градусов

Вводится число N, задающее размер матрицы NxN. Далее вводятся элементы массива построчно. В строке элементы разделены пробелом. Необходимо выполнить преобразование массива и поменять местами строки со столбцами.

Формат входных данных

Целое положительные числа N >= 2 N строк содержащие N целых чисел

Формат выходных данных

Двумерный массив, разделитем является символ табуляции "\t"

Примеры:

Входные данные

2

1 2

3 4

Результат работы

1 3

2 4

Задание 9

Меняем самую короткую и длинную строку в массиве

Вводится число N - количество строк в двумерном массиве. Далее вводятся элементы массива построчно. В строке элементы разделены пробелом. Необходимо выполнить преобразование массива и поменять местами строку, в которой наименьшее количество элементов со строкой, в которой наибольшее количество элементов. Если строк с минимальной/максимальной строкой несколько, то использовать первую строку с минимальной/максимальной длиной. Если все строки одинаковы, то поменять местами первую и последнюю строку в массиве.

Формат входных данных

Целое положительное число N >= 2 N строк содержащее целые числа

Формат выходных данных

Двумерный массив, разделитем является символ табуляции "\t"

Примеры:

Входные данные

4

1

2 2

3 3 3

4 4 4

Результат работы

3 3 3

2 2

1

4 4 4

Входные данные

3

1 2 3

4 5 6

7 6 9

Результат работы

7 6 9

4 5 6

1 2 3

Задание 10

Заполнение массива змейкой

Вводятся два числа разделенных пробелом N и M. Необходимо создать матрицу N x M и заполнить её змейкой. Первый элемент массива равен единице, второй элемент 2 и т.д.

Формат входных данных

Целые положительные числа N и M >= 2

Формат выходных данных

Двумерный массив, разделителем является символ табуляции "\t"

Примеры:

Входные данные

4 5

Результат работы

1 2 3 4 5

10 9 8 7 6

11 12 13 14 15

20 19 18 17 16

Задание 11

Заполнение массива спиралью

Вводятся два числа разделенных пробелом N и M. Необходимо создать матрицу N x M

и заполнить её по спирали, выходящей из левого верхнего угла и закрученной по часовой стрелке, как показано в примере. Первый элемент массива равен единице, второй элемент 2 и т.д.

Формат входных данных

Целые положительные числа N и M >= 2

Формат выходных данных

Двумерный массив, разделителем является символ табуляции "\t"

Примеры:

Входные данные

4 5

Результат работы

1	2	3	4	5
14	15	16	17	6
13	20	19	18	7
12	11	10	9	8

Входные данные

3 3

Результат работы

1	2	3
8	9	4
7	6	5

Лабораторная работа №2. Обобщения, коллекции, интерфейсы коллекций, классы коллекций, итераторы, отображения (1 ч)

Задание 1

Перед нами поставлена задача реализовать основной алгоритм работы разменного аппарата. Суть аппарата простая: пользователь вводит сумму, со счета банковской карты оплачивает ее, а аппарат выдает ему запрошенную сумму в виде набора монет. Для упрощения будем считать, что количество монет в аппарате бесконечно.

Требования к работе:

- Аппарат должен выдавать минимально возможное количество монет;
- Аппарат должен при создании получать номинал монет, с которыми он будет работать и корректно работать с любым набором монет;
- Аппарат должен проверять возможен ли размен введенной суммы существующим набором монет;
- Аппарат должен проверять все ситуации, в которых размен не возможен и выдавать соответствующее исключение с указанием конкретной причины причины отказа.

Вам необходимо прокомментировать ваш код, чтобы раскрыть логику работы. Также вам необходимо реализовать программу, в которой вы наглядно продемонстрируете работу вашего аппарата с различными входными данными.

Пример

- Введенная сумма 15, аппарат работает с набором монет: 1, 5, 10, 25, 100. Аппарат должен вернуть одну монету номиналом 5, одну монету номиналом 10.
- Введенная сумма 40, аппарат работает с набором монет: 1, 5, 10, 25, 100. Аппарат должен вернуть по одной монете номиналом 5, 10, 25.

Задание 2

В магазине проходит акция на продажу популярной серии из 5 книг. Для стимулирования продаж предлагается скидка на покупку нескольких книг из данной серии.

Условия акции:

- Один экземпляр из пяти книг стоит 400 рублей.
- Если вы покупаете две разные книги, вы получаете скидку 5% на эти две книги.
- Если вы покупаете 3 разных книги, вы получаете скидку 10%.
- Если вы покупаете 4 разных книги, вы получаете скидку 20%.
- Если вы купите все 5, вы получите скидку 25%.

Обратите внимание: если вы купите четыре книги, из которых три имеют разные названия, а четвертая дублируют одну из этих трех, то вы получите скидку 10% на три книги,

входящие в комплект, но четвертая книга по-прежнему стоит 400 рублей.

Ваша задача состоит в том, чтобы написать класс BookStore для расчета цены любой мыслимой корзины для покупок (содержащей только книги этой серии), предоставляя как можно большую скидку.

Пример:

Имеется следующий набор:

- 2 экземпляра первой книги;
- 2 экземпляра второй книги;
- 2 экземпляра третьей книги;
- 1 экземпляр четвертой книги;
- 1 экземпляр пятой книги.

Один из способов расчета для этого набора:

- 1 группа из 5 -> 25% скидка (1, 2, 3, 4, 5)
- +1 группа из 3 -> 10% скидка (1, 2, 3)

Это даст в общей сложности:

- 5 книг со скидкой 25%
- +3 книги со скидкой 10%

В результате чего:

- 5 x (400 - 100) = 5 x 300 = 1500 руб;
- +3 x (400 - 40) = 3 x 360 = 1080 руб.

На общую сумму 2580 руб.

Однако есть другой способ сгруппировать эти 8 книг:

- 1 группа из 4 книг -> 20% скидка (1, 2, 3, 4)
- +1 группа из 4 книг -> 20% скидка (1, 2, 3, 5)

Это даст в общей сложности:

- 4 книги со скидкой 20%
- +4 книги со скидкой 20%

В результате чего:

- 4 x (400 - 80) = 4 x 320 = 1280 руб.
- +4 x (400 - 80) = 4 x 320 = 1280 руб.

На общую сумму 2560 руб.

А 2560 рублей - это цена с самой большой скидкой.

Реализуйте метод calculateBasketCost, принимающий список покупаемых книг в виде List<Integer>

Задание 3

Реализуйте двусвязный список. Как и массив, связанный список представляет собой простую линейную структуру данных. Несколько общих типов данных могут быть реализованы с использованием связанных списков, таких как очереди, стеки и ассоциативные массивы. Вы пишите реализацию двусвязного списка. Элемент списка содержит значение и указателей на следующую и предыдущую элемент списка. Также список должен содержать ссылки на первый и последний узел и предлагать удобный интерфейс для добавления и удаления элементов:

- push (вставить запись в конец списка);
- pop (убрать последнюю запись в списке);
- shift (убрать первую запись);
- unshift (вставить запись спереди).

Это упражнение знакомит с дженериками. Чтобы сдача задания прошла успешно, вам нужно построить свой класс таким образом, чтобы он принимал любые типы данных, например, Integer или String.

Лабораторная работа №3. Бинарный поиск (1 ч)

1. Напишите метод, который проверяет, входит ли в массив заданный элемент или нет. Используйте перебор и двоичный поиск для решения этой задачи. Сравните время выполнения обоих решений для больших массивов (например, 100000000 элементов).

2. Реализовать бинарное дерево поиска:

Реализовать структуру данных "Дерево". Необходимо добавить функции:

1) добавление элемента в дерево;

2) удаление элемента из дерева;

3) поиск элемента в дереве;

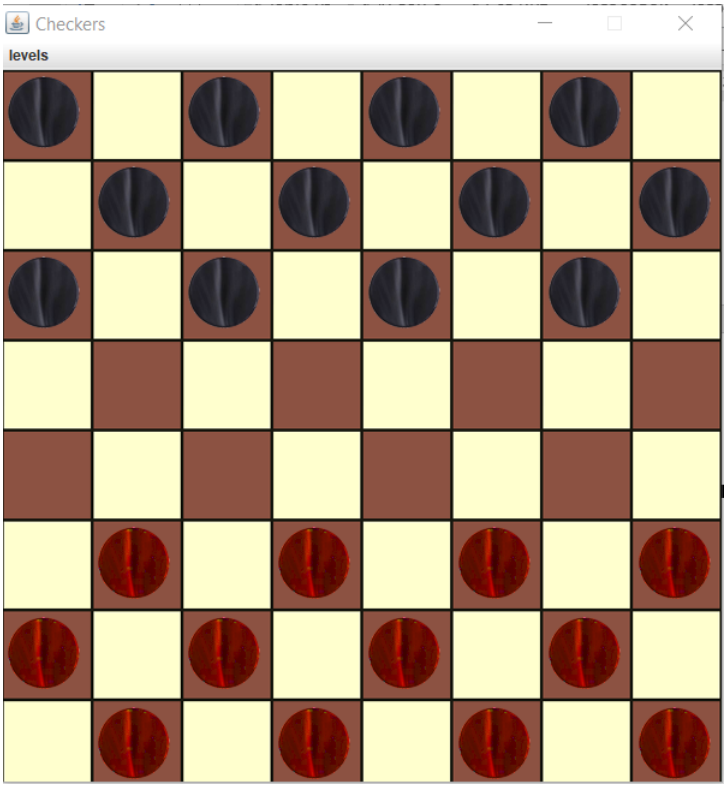
4) поиск максимального и минимального элемента в дереве;

5) вывод дерева на экран в виде иерархии.

Лабораторная работа №4. Графический интерфейс java-приложений. Swing и AWT (3 ч)

Задание для самостоятельного выполнения

Реализовать игру в шашки, используя библиотеки *AWT* и *Swing*. Пример интерфейса приведен ниже.



РАЗДЕЛ 2. ВВЕДЕНИЕ В SPRING FRAMEWORK

Лабораторная работа №5. Разработка приложений с использованием Java Server Pages (JSP) (5 ч)

Задание для самостоятельного выполнения

Необходимо создать JSP-страницу, реализующую страницу списка продуктов интернет-магазина. Основные требования:

1.Список продуктов реализован с использованием JSP.

2.Список обеспечивает фильтрацию продуктов с использованием JavaScript и метода GET.

3.Из списка есть возможность перехода к карточке продукта (лабораторная №1).

4.Элементы списка формируются в виде JavaBean.

5.Элементы списка отображаются с помощью вложенных (include) страниц. При этом для каждого элемента отображается рисунок, название, краткие характеристики и кнопка «Купить» (реализация появится после создания «Корзины»).

6.На списке есть и реализована кнопка локализации для трёх языков (обязательные: русский, английский).

7.Оформление списка выполнено с использованием CSS.

8.В «шапке» списка находится вложенная (include) страница, содержащая информацию о пользователе со следующими кнопками: «Вход», «Корзина», «История покупок» и кнопка локализации (реализация будет после выполнения соответствующих лабораторных работ).

9.Последние значения фильтра сохраняются в Cookie. При первой загрузке страницы автоматически применяются значения фильтров, сохранённые в Cookie.

Лабораторная работа №6. Работа с Spring MVC (5 ч)

Задания для самостоятельного выполнения

Создать 3 XML-файла. Первый файл должен содержать сведения о книгах (**код книги, название, код автора, жанр, возрастное ограничение**), второй – об авторах (**код автора, фио**), третий – о читателях (**код читателя, фио, дата рождения, телефон, дата получения книг(и), взятые книги – код книги**).

Структура третьего XML-файла.

<Читатели>

<Читатель код_читателя = «»>

<ФИО></ФИО>

<Дата_рождения><\Дата_рождения>

<Телефон><\Телефон>

<Дата_получения_книги><\Дата_получения_книги>

<Книги>

<Код книги><\Код книги>

<\Книги>

</Читатель>

<\Читатели>

1. Создать приложение, которое позволит выполнять добавление, извлечение и поиск информации из XML-файлов. Предусмотреть редактирование и добавление новых данных о книгах и читателях. При добавлении данных о читателе и взятых книгах проверять возраст читателя, при несоответствии возрастной категории не добавлять и выводить сообщение.
2. Приложение должно осуществлять поиск книг по названию, автору и жанру.
3. Осуществлять поиск по ФИО читателей.
3. Вывести все книги по каждому автору.
4. Приложение должно осуществлять вывод информации о взятых книгах выбранного читателя (название книги, автор, жанр).
5. Определить количество читатель за выбранный промежуток времени.

Вывести самого заядлого читателя.

Создайте веб-сайт, используя фреймворк Spring MVC.

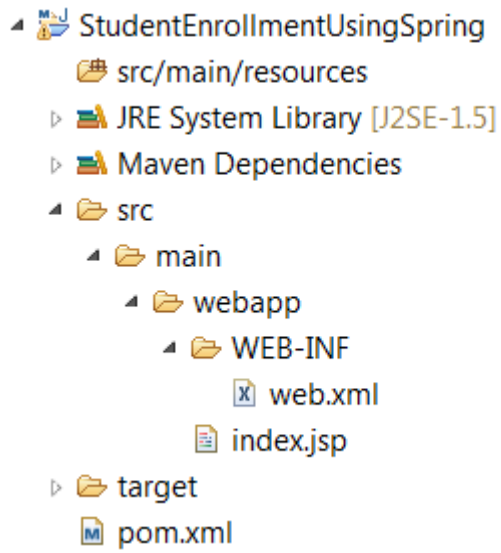
Лабораторная работа №7. Создание веб-приложения на Java с использованием Hibernate с Spring (6 ч)

Задания для самостоятельного выполнения

Разработать веб-приложение, которое позволяет управлять информацией о клиентах с помощью PostgreSQL DB с Hibernate ORM в среде Spring. Это простое приложение, которое предназначено для сбора входных данных от пользователя во время регистрации, сохранения данных в базе данных PostgreSQL и аутентификации при входе.

1. Создание проекта веб-приложения Java с использованием шаблона Maven.

Для начала в IDE создайте проект Java Maven с шаблоном maven-archetype-webapp (отфильтруйте каталог на основе строки «webapp»), указав соответствующие значения для GroupId и Artifact Id для проекта. Пример структуры каталогов веб-приложения показан ниже со стандартным дескриптором развертывания web.xml и Maven pom.xml.



2. Обновите pom.xml

Чтобы вышеуказанный проект Maven Java Web Application поддерживал Hibernate ORM в среде Spring, добавьте следующие зависимости в существующий файл pom.xml:

jstl, spring-webmvc и servlet-api (для поддержки Spring)

mysql-connector-java (для поддержки MYSQL)

spring-jdbc (для доступа к данным с помощью JDBC Spring)

Spring-Orm (для доступа к данным ORM с помощью Spring)

spring-data-jpa (для поддержки JPA)

hibernate-validator и hibernate-entitymanager (для поддержки Hibernate)

JTA (для поддержки транзакций)

```
<dependency>
```

```
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>3.2.4.RELEASE</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.21</version>
</dependency>
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-validator</artifactId>
  <version>4.2.0.Final</version>
</dependency>
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>4.1.9.Final</version>
</dependency>
<dependency>
  <groupId>javax.transaction</groupId>
  <artifactId>jta</artifactId>
  <version>1.1</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>3.2.0.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>3.2.0.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-jpa</artifactId>
  <version>1.3.0.RELEASE</version>
  <exclusions>
      <exclusion>
          <groupId>org.springframework</groupId>
          <artifactId>spring-aop</artifactId>
      </exclusion>
  </exclusions>
</dependency>
```

3. Изменить web.xml

Измените содержимое файла web.xml, включив в него следующее:

- Сервлет и указать расположение файла конфигурации для того же. В этом примере файл конфигурации с именем springConfig.xml создается в папке WEB-INF / config в макете проекта.
- Отображение сервлета для сопоставления сервлета, созданного на предыдущем шаге, который должен вызываться, когда клиент указывает URL-адрес, соответствующий шаблону URL-адреса.
- ContextLoaderListener для интеграции Spring с веб-приложением и предоставления contextConfigLocation, где находятся файлы контекста для JPA.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

<servlet>
  <servlet-name>studentHibernateServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/config/servletConfig.xml</param-value>
  </init-param>
</servlet>

<servlet-mapping>
  <servlet-name>studentHibernateServlet</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:/jpaContext.xml</param-value>
</context-param>

<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

  <display-name>Archetype Created Web Application</display-name>
</web-app>
```

4. Создайте файл конфигурации Spring

Создайте файл конфигурации Spring Bean в папке WEB-INF / config. Если STS (Spring Tool Suite) является IDE, включите пространство имен context и mvc. ServletConfig.xml будет таким, как показано ниже:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd
      http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
```

```
      http://www.springframework.org/schema/context http://www.springframework
.org/schema/context/spring-context-3.2.xsd">

</beans>
```

После включения необходимых пространств имен добавьте следующее (между тегами <beans> и </ beans>), чтобы указать, что приложение управляется аннотациями и является базовым пакетом для сканирования компонента контекста.

```
<mvc:annotation-driven />

<context:component-scan base-package="com.github.elizabetht" />
```

Включите bean-компонент InternalResourceViewResolver для Spring, чтобы найти файлы jsp

```
<bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix" value="/WEB-INF/jsp/" />
  <property name="suffix" value=".jsp" />
</bean>
```

Включите компонент для указания файла свойств (подробнее об этом позже), который будет использоваться для хранения пользовательских сообщений или свойств. Следующая конфигурация позволяет создать файл свойств с именем messages.properties в папке src / main / resources в проекте.

```
<bean id="messageSource"
class="org.springframework.context.support.ResourceBundleMessageSource">
  <property name="basename" value="messages" />
</bean>
```

5. Создайте файл persistence.xml

Создайте файл с именем persistence.xml в папке src / main / resources / META-INF в проекте, чтобы определить единицу сохранения, требуемую JPA. Добавьте следующее в файл persistence.xml, чтобы определить единицу хранения с именем punit.

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="{http://java.sun.com/xml/ns/persistence}
{http://java.sun.com/xml/ns/persistence_2_0.xsd}"
  version="2.0">

  <persistence-unit name="punit">
  </persistence-unit>

</persistence>
```

6. Создайте jpaContext.xml

Как определено в файле web.xml, создайте файл с именем jpaContext.xml в папке src / main / resources в проекте, чтобы определить конфигурации, связанные с JPA и Hibernate.

Обратите внимание, что любой файл, созданный в папке src / main / resources в проекте maven, будет автоматически добавлен Maven в путь к классам. Если STS (Spring Tool Suite) является IDE, включите пространства имен context, jpa и tx. Файл jpaContext.xml будет таким, как показано ниже:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:jpa="http://www.springframework.org/schema/data/jpa"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
      http://www.springframework.org/schema/data/jpa http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd
      http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
      http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd">

<beans>
```

После включения необходимых пространств имен добавьте следующее (между тегами <beans> и </ beans>), чтобы указать, что приложение управляется аннотациями и является базовым пакетом для сканирования репозиториев jpa.

```
<context:annotation-config />

<jpa:repositories base-package="com.github.elizabetht.repository" />
```

Затем включите bean-компонент PersistenceAnnotationBeanPostProcessor. Это необходимо для обработки модуля сохраняемости, аннотаций постоянства и для ввода ресурсов, связанных с JPA.

```
<bean class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor" />
```

Включите компонент для EntityManagerFactory, в котором перечислены различные свойства / ресурсы, связанные с JPA.

```
<bean id="entityManagerFactory"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
      <property name="persistenceUnitName" value="punit" />
      <property name="dataSource" ref="dataSource" />
      <property name="jpaVendorAdapter">
          <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
              <property name="showSql" value="true" />
          </bean>
      </property>
      <property name="jpaPropertyMap">
          <map>
              <entry key="hibernate.dialect"
value="org.hibernate.dialect.MySQL5InnoDBDialect" />
              <entry key="hibernate.hbm2ddl.auto" value="validate" />
              <entry key="hibernate.format_sql" value="true" />
          </map>
```

```
    </property>
</bean>
```

Включите bean-компонент для источника данных, где можно указать свойства базы данных PostgreSQL, такие как url, имя пользователя и пароль. Замените <include connection url> фактическим URL-адресом соединения для подключения к базе данных PostgreSQL. Аналогичным образом замените <include username> и <include password> действительными значениями имени пользователя и пароля.

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="com.mysql.jdbc.Driver" />
      <property name="url" value="jdbc:mysql//<include connection
url>:3306/studentEnrollment?autoReconnect=true&createDatabaseIfNotExist=true&" />
      <property name="username" value="<include username>" />
      <property name="password" value="<include password>" />
</bean>
```

Включите bean-компонент для менеджера транзакций для определения объема / управления транзакциями.

```
<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
      <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>

<tx:annotation-driven transaction-manager="transactionManager" />
```

7. Создайте файлы JSP для регистрации / входа ученика

Создайте папку с именем «jsp» в WEB-INF (именно здесь будут созданы файлы jsp, как указано в servletConfig.xml для bean-компонента InternalResourceViewResolver).

Создайте файл signup.jsp, чтобы включить форму для получения данных ввода, таких как Имя пользователя, Пароль, Имя, Фамилия, ДатаОfBirth и Адрес электронной почты студента.

Затем создайте файл login.jsp для включения формы с именами пользователей и паролями.

Также создайте success.jsp, чтобы указать успешный вход в систему, и fail.jsp, чтобы указать сбой входа в систему (это просто страницы, используемые для отображения содержимого — логика обработки не используется).

8. Создание пакетов для классов уровня Controller, Model, Repository и Service

Создайте пакеты для классов Spring Controller, Model, Repository и Service в папке src / main / java.

9. Создайте классы для уровня модели

Создайте класс POJO с именем Student.java внутри пакета com.github.elizabetht.model, чтобы включить детали сущности модели Student во время регистрации. Создайте еще один класс POJO с именем StudentLogin.java внутри того же пакета com.github.elizabetht.model, чтобы включить данные для входа в систему.

10. Создайте класс для уровня репозитория

Создайте интерфейсный класс StudentRepository.java внутри пакета com.github.elizabetht.repository для поддержки операций базы данных уровня репозитория.

Для целей приложения необходимы два метода интерфейса:

- Чтобы вставить данные регистрации ученика в базу данных
- Чтобы проверить регистрационные данные студента из базы данных

```
@Repository("studentRepository")
public interface StudentRepository extends JpaRepository<Student, Long> {

  @Query("select s from Student s where s.userName = :userName")
  Student findByUserName(@Param("userName") String userName);

}
```

Метод save () поддерживается реализацией Hibernate, поэтому для вставки данных не требуется никаких отдельных операторов SQL.

11. Создайте классы для уровня обслуживания

Создайте интерфейсный класс с именем StudentService.java внутри пакета com.github.elizabetht.service для поддержки операций уровня обслуживания.

Создайте класс реализации уровня сервиса (действительно POJO) с именем StudentServiceImpl.java внутри пакета com.github.elizabetht.service. Именно в этом заключается логика приложения — либо для сохранения сведений об ученике в базе данных, либо для проверки (уже сохраненных) сведений об ученике из базы данных.

12. Создайте класс для уровня контроллера

Создайте класс POJO уровня контроллера с именем StudentController.java внутри пакета com.github.elizabetht.controller. Именно здесь идет логика маршрутизации приложения — вызывается ли действие регистрации или входа в систему.

13. Создайте файл messages.properties

аннотация @Valid используется для проверки входных параметров формы, достигающей метода, и результат проверки сохраняется в объекте BindingResult. Для проверки конкретных полей используйте аннотации как @NotEmpty, @Size, @Email и @NotNull из различных проверок, доступных в Hibernate Validator.

Пользовательские сообщения, которые должны отображаться при сбое любого из вышеупомянутых валидаторов, указываются в файле messages.properties. Создайте файл с именем messages.properties в папке src / main / resources и включите следующее:

```
NotEmpty=Field cannot be blank
NotNull=Field cannot be blank

Email=Email Address not valid/well-formed
Past=Date of Birth must be in the past

Size={0} must be between {2} and {1} characters long
typeMismatch=Invalid format
```

14. Создайте схему БД в базе данных PostgreSQL

Подключитесь к базе данных PostgreSQL, которая будет использоваться для этого приложения, и создайте новую схему базы данных с именем studentEnrollment. Это необходимо, поскольку имя схемы БД studentEnrollment указывается в bean-компоненте dataSource в jpaContext.xml.

После создания схемы DB studentEnrollment создайте таблицу с именем student внутри схемы DB.

15. Развертывание приложения

Веб-приложение Java можно развернуть локально, щелкнув правой кнопкой мыши по проекту и выбрав опцию «Запуск от имени -> Запуск на сервере».

РАЗДЕЛ 3. РАЗРАБОТКА ВЕБ-СЕРВИСА

Лабораторная работа №8. Spring и работа с БД. (5 ч)

Изучить:

https://habr.com/ru/post/333756/

https://habr.com/ru/post/435114/

https://habr.com/ru/post/320542/

https://habr.com/ru/post/265061/

https://www.youtube.com/watch?v=r2Yi6Xc7UwE

https://devcolibri.com/spring-data-jpa-%D0%BF%D0%B8%D1%88%D0%B5%D0%BC-dao-%D0%B8-services-%D1%87%D0%B0%D1%81%D1%82%D1%8C-2/

https://www.youtube.com/watch?v=nyFLX3q3poY&t=515s

https://habr.com/ru/company/haulmont/blog/440696/

https://habr.com/ru/post/438870/

https://www.baeldung.com/orika-mapping

https://www.youtube.com/watch?v=VSNZN9wJlaQ

https://dzone.com/articles/spring-boot-2-restful-api-documentation-with-swagg

Выполните следующие задания:

1. Создать проект “currency-api” используя spring boot

2. Подтянуть необходимые spring-boot зависимости и плагины используя gradle.

3. Проект должен иметь следующую структуру:

- ru.ds.education.currency.model (будет содержать entity)

- ru.ds.education.currency.repository (будет содержать spring data репозитории)

- ru.ds.education.currency.service (будет содержать сервисный слой)

- ru.ds.education.currency.controller (будет содержать непосредственно контролеры, предоставляющие api)

- ru.ds.education.currency (будет содержать конфигурации и исполняющий класс)

4. Создать таблицу

```
curs_data (
  id bigint,
  currency varchar(3),
  curs number(5,2),
  curs_date date)
```

Для создания таблицы использовать скрипты миграции.

Названия скриптов должны иметь формат
V{YYYY}_{MM}_{DD}_{hh}_{mm}{кратко бизнес смысл}.sql, например
V2018_07_05_19_00init.sql или
V2018_10_23_19_00_alter_application_create_date_local.sql

5. Описать entity к таблице. Наименование всех валют зафиксировать в ENUM. Поле в entity, которое описывает валюту в бд – должно быть ENUM;

6. Создать ещё одну миграцию, которая добавить в curs_data данные по курсам валют за 3 дня;

7. Создать репозитории;

8. Создать сервисы – сервисы в себе должны инджектить репозитории. Обращение к репозиториям будет только внутри сервисов;

9. Создать контроллеры. Контроллер будет обращаться к сервисам. Контроллеры должны предоставлять CRUD Api (GET, PUT, POST, DELETE), а также api по получению курса валюты по коду валюты и дате;

10. Отдавать Entity в REST напрямую запрещено, необходимо преобразование в DTO в контроллере, использовать Orika Mapper;

11. Для создания гетеров и сетеров использовать lombok;

12. Создать описание сервиса с помощью swagger. Аннотировать контроллеры, DTO и поля DTO. В описание включить бизнес логику;

13. Протестировать работу сервиса с помощью API клиента (Например, Postman, Insomnia)

Лабораторная работа №9. Messaging. JMS. Тестирование (5 ч)

Изучить:

https://www.youtube.com/watch?v=6aK7ZBrjH-Q

https://www.youtube.com/watch?v=5ITop8rjHbk&t=191s

https://itnext.io/event-driven-microservices-with-spring-boot-and-activemq-5ef709928482

https://habr.com/ru/post/169381/

https://habr.com/ru/post/120101/

https://habr.com/ru/post/337700/

Дополнительно:

https://habr.com/ru/company/jugru/blog/329372/

https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-testing

Выполните следующие задания:

- Создать проект “currency-cbr-adapter”;
- Подтянуть необходимые spring-boot зависимости и плагины используя gradle;
- Модуль будет представлять из себя адаптер интеграции между ЦБ и нашей средой;
- Модуль с одной стороны слушает очередь, с другой вызывает сервисы ЦБ;
- В модуле необходимо реализовать клиента веб сервиса ЦБ. Описание сервиса http://www.cbr.ru/DailyInfoWebServ/DailyInfo.asmx;
- Для реализации адаптера необходимо использовать только одну операцию веб сервиса ЦБ - “GetCursOnDateXML”;
- Классы модели данных клиента, а так-же прокси сервисы клиента должны генерироваться кодогенератором. Для генерации можно использовать, например, gradle плагин “no.nils.wsdl2java”;
- Модуль будет слушать очередь dev.cbr.request, а ответ класть в очередь dev.cbr.response;
- Формат данных в сообщениях – json;
- Формат запроса: {"onDate": "string(YYYY-MM-DD)"};
- Формат ответа: {"onDate": "string(YYYY-MM-DD)", "rates" : [{"currency": "string", "curs": decimal}]};
- При получении сообщения из очереди запроса необходимо получить correlationId заголовок. Установить его в ответном сообщении, если он имелся;

- По итогу приложение должно считывать запрос из очереди dev.cbr.request, отправлять запрос на сервисы ЦБ, возвращать ответ в очередь dev.cbr.response;

- Модуль обложить тестами запроса-ответа.

Лабораторная работа №10. Spring Async (6 ч)

Изучить:

https://spring-projects.ru/guides/async-method/

https://www.baeldung.com/spring-async

https://dzone.com/articles/spring-boot-creating-asynchronous-methods-using-as

Выполните следующие задания

Доработка currency-api и интеграция с currency-cbr-adapter:

- Добавить необходимые зависимости в gradle;
- добавить таблицу curs_request (
  id bigint, - идентификатор в бд
  curs_date date, - дата курса валюты
  request_date timestamp, - дата и время создания запроса
  correlation_id varchar, - идентификатор сообщения запроса
  status varchar - статус запроса);
- добавить sequence curs_request_seq. При создании записей из ORM, Id записи должно заполняться из sequence;
- Все изменения в БД должны осуществляться через скрипты миграции;
- Добавить entity для curs_request.
- Status может принимать значения: CREATED, SENT, PROCESSED, FAILED;
- Обновить логику модели при запросе в api курса валюты на дату следующим образом:
  - проверяется, есть запись в таблице curs_data. Если есть, то возвращается ответ со статусом 200 и телом ответа;
  - Если нет записи в таблице curs_data, то проверяется есть ли запись в таблице curs_request. Выборка из таблицы осуществляется по curs_date, и по максимальной request_date. Если есть запись в таблице curs_request и статус ее не равен FAILED, то REST API должно вернуть ответ со статусом 202 и с пустым телом;
  - Если нет записи в таблице curs_request или статус FAILED, то создается новая запись. Поле статус заполняется значением CREATED. Далее должен отправиться запрос в очередь запросов (dev.cbr.request). Как только сообщение отправлено в очередь, статус у соответствующей записи необходимо изменить на SENT. Далее необходимо подключиться и слушать очередь dev.cbr.response. Вычитывать сообщение необходимо по correlationId. Соответственно, при отправке его нужно проставить. Проставляется обычно рандомным uuid-ом;

- Таймаут на вычитывание, должен быть 10 сек и конфигурироваться в файле конфигураций;

- Если ответ вернулся успешный, то необходимо из ответа создать записи в таблице curs_data, а если какие либо записи существуют, то обновить их;

- В таблице curs_request, в соответствующей записи, проставляется статус PROCESSED.

- Если произошла ошибка при считывании, либо таймаут, то соответствующей записи проставляется статус FAILED. При чем этот пункт должен выполнится в своем потоке (ассинхронно, см. spring async), т.е. должен запуститься асинхронный метод, и api должно вернуть ответ со статусом 202 и пустым телом.

- При всем этом, связка отправки запроса и считывания ответа должна выполнятся в отдельном потоке ассинхронно (см spring async). Т.е. выборки из бд, должны происходить в разных потоках;

- Для взаимодействия с очередями использовать jmstemplate (см. convertandsend, receiveandconvert)

РАЗДЕЛ 4. РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНЫХ ПРИКЛАДНЫХ ПРОГРАММ ДЛЯ ОБРАБОТКИ ЕСТЕСТВЕННОГО ЯЗЫКА И РАСПОЗНАВАНИЯ РЕЧИ

Лабораторная работа №13. Реализация алгоритма «Вперед-Назад» (7 ч)

Цель лабораторной работы: изучение алгоритма «Вперед-Назад», реализация прямого и обратного алгоритма в скрытой марковской модели.

Скрытая марковская модель - это цепь Маркова, которая в основном используется в задачах с временной последовательностью данных. Модель Маркова объясняет, что следующий шаг зависит только от предыдущего шага во временной последовательности.

Скрытая марковская модель (θ) имеет следующие параметры:

Набор из M скрытых состояний (S^M)
Матрица вероятности транзакции (A)
Последовательность T наблюдений (V^T)
Матрица вероятности выбросов (также известная как вероятность наблюдения) (B)
Начальное распределение вероятностей (π)

Алгоритм вперед – назад - это обычный рекурсивный и эффективный способ оценки, скрытой марковской модели, то есть вычисления вероятности последовательности наблюдений при данной модели. Эта вероятность может использоваться для классификации последовательностей наблюдений в приложениях распознавания.

В прямом алгоритме используется вычисленная вероятность на текущем временном шаге, чтобы получить вероятность следующего временного шага.

Учитывая последовательность видимого состояния V^T, какова будет вероятность того, что скрытая марковская модель окажется в определенном скрытом состоянии s на определенном временном шаге t.

Обратный алгоритм - это обращенная во времени версия прямого алгоритма. В обратном алгоритме нам нужно найти вероятность того, что машина будет в скрытом состоянии s на временном шаге t и будет генерировать оставшуюся часть последовательности видимого символа V^T.

Данные:

Есть 2 скрытых состояния (A, B) и 3 видимых состояния (0,1,2).

$$A = \begin{bmatrix} 0{,}54 & 0{,}46 \\ 0{,}49 & 0{,}51 \end{bmatrix}$$

$$B = \begin{bmatrix} 0{,}16 & 0{,}26 & 0{,}58 \\ 0{,}25 & 0{,}28 & 0{,}47 \end{bmatrix}$$

В data.csv имеет два столбца с именами Hidden и Visible.

Задание на лабораторную работу: Необходимо реализовать прямой и обратный алгоритм в скрытой марковской модели.

Лабораторная работа №14. Сверточные коды и алгоритм декодирования Витерби (7 ч)

Цель лабораторной работы: изучение алгоритма декодирования Витерби.

Задание на лабораторную работу: в соответствии с номером варианта требуется закодировать информационную последовательность битов с использованием сверточного кода, а также произвести декодирование принятой кодовой комбинации по алгоритму Витерби, определить наличие ошибок в принятой кодовой комбинации и исправить их.

Задание на выполнение лабораторной работы согласно номеру варианта

Номер варианта	Входной информационный сигнал	Принятый сигнал
1	11010110100010	11011111001011110100011111010111011111101
2	10010110101010	11011111101011010100011111011100001011101
3	01011110101010	11011111001000101000011110011100000001101
4	11110110110011	11011111101010101000011111010000011001101
5	10001110101010	11111111101000101000010100001101110011101
6	10010010000011	11011011101011000011011100001101100011101
7	11111110101000	11011111100010110011011100000101110011101
8	00010110101111	11011110100110110011010001111101110011001
9	10011111101110	11010111010001110011010001111100110011101
10	10110001101001	11011111010010101011010001111100110011101
11	10101101100110	11011110010010101111100010111101110011101
12	10000100001011	11010111010010101111100110111101001100011
13	10110111101011	11011111010000100010010110110101000100011
14	10010111111011	11011110010010100010010110111110011100011
15	11110000101110	11011101010010100010010110111100010101110

Рисунок 1

1. Для сверточного кодера, схема которого приведена на рис. 2, требуется закодировать информационную последовательность битов на входе кодера с целью получения выходного сигнала – сверточного кода.

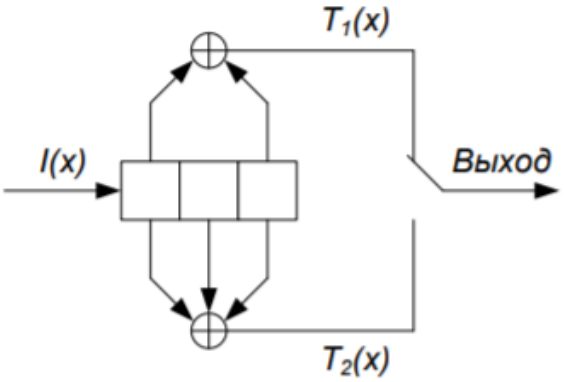


Рисунок 2

2. Написать функцию, реализующую декодирование входных данных декодером, построенному по алгоритму Витерби.

3. В результате выполнения данного этапа студенту требуется найти ошибки в полученной кодовой комбинации, указать их позиции и исправить.

Лабораторная работа №15. Анализ текста как источника больших данных. (4 ч)

Анализ текста состоит из следующих модулей:

– Графематический анализ (сегментация), т. е. выделение в тексте предложений и словоформ, точнее токенов (т. к. в тексте могут быть не только слова) — переход от символов к словам;

– Морфологический анализ — переход от словоформ к их леммам (словарным формам лексем) или основам (ядерным частям слова, за вычетом словоизменительных морфем);

– Синтаксический анализ — выявление синтаксических связей слов и грамматической структуры предложений;

– Семантический и прагматический анализ, при котором определяется смысл фраз и соответствующая реакция системы.

Цель лабораторной работы: познакомиться с библиотеками NLP.

Задание на лабораторную работу:

1. Выбрать вырезку из статьи по машинному обучению.
2. Установить библиотеку NLP для работы с текстом.
3. Провести предварительную обработку текста по приведенным ниже этапам.

Очистка текстовых данных необходима, чтобы выделить атрибуты, которые мы хотим использовать в нашей системе машинного обучения. Очистка (или предварительная обработка) данных обычно состоит из нескольких этапов:

– ***Удалить пунктуацию***

Пунктуация может обеспечить грамматический контекст для предложения, которое поддерживает наше понимание. Но для нашего векторизатора, который считает количество слов, а не контекст, он не добавляет значения, поэтому мы удаляем все специальные символы.

– ***Tokenization***

Токенизация разделяет текст на блоки, такие как предложения или слова. Это дает структуру ранее неструктурированному тексту.

– Удалить стоп-слова

Стоп-слова - это обычные слова, которые, скорее всего, появятся в любом тексте. Они мало говорят нам о наших данных, поэтому мы их удаляем.

– ***Stemming***

Стемминг помогает свести слово к его форме. Часто имеет смысл рассматривать похожие слова одинаково. Он удаляет достаточно, например, «ing», «ly», «s» и т. Д. Простым подходом, основанным на правилах. Это уменьшает корпус слов, но часто фактическими

словами пренебрегают.

– ***Лемматизация***

Лемматизация выводит каноническую форму («лемму») слова. т.е. корневая форма. Это лучше, чем основа, так как он использует подход на основе словаря, то есть морфологический анализ к корню. Stemming обычно быстрее, так как он просто отсекает конец слова, не понимая контекста слова. Лемматизация происходит медленнее и точнее, так как требует осознанного анализа с учетом контекста слова.

Векторизация данных

Векторизация - это процесс кодирования текста в виде целых чисел, то есть числовой формы для создания векторов признаков, чтобы алгоритмы машинного обучения могли понимать наши данные.

– ***Мешок слов***

Bag of Words (BoW) или CountVectorizer описывает наличие слов в текстовых данных. Это дает результат 1, если присутствует в предложении и 0, если не присутствует. Поэтому в каждом текстовом документе создается пакет слов с количеством матриц документов.

– ***TF-IDF***

Вычисляет «относительную частоту» появления слова в документе по сравнению с его частотой во всех документах. Это более полезно, чем термин «частота» для определения «важных» слов в каждом документе.

Заметка: Используется для оценки в поисковых системах, суммирования текста, кластеризации документов.

Методические указания рекомендованы на заседании МКН подготовки 09.04.02 «Информационные системы и технологии» профиль «Искусственный интеллект в проектировании городской среды» к размещению на образовательном портале ГАОУ АО ВО «АГАСУ» (http://moodle.aucu.ru)