МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

УТВЕРЖДАЮ:
Директор института
_____ С.И. Лукьянов
«__» _____ 2017 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

ОСНОВЫ МАШИННОГО ОБУЧЕНИЯ

Направление подготовки
09.03.01 Информатика и вычислительная техника

Профиль программы
Автоматизированные системы обработки информации и управления

Уровень высшего образования – бакалавриат

Программа подготовки – прикладной бакалавриат

Форма обучения
Очная

Институт	*энергетики и автоматизированных систем*
Кафедра	*вычислительной техники и программирования*
Курс	*4*
Семестр	*7*

Магнитогорск
2017 г.

Рабочая программа составлена на основе ФГОС ВО по направлению подготовки (специальности) 09.03.01 Информатика и вычислительная техника, утвержденного приказом МО и Н РФ от 12.01.2016 № 5.

Рабочая программа рассмотрена и одобрена на заседании кафедры вычислительной техники и программирования «26» апреля 2017 г., протокол № 2.

Зав. кафедрой _____ / О.С. Логунова/

Рабочая программа одобрена методической комиссией института энергетики и автоматизированных систем «27» апреля 2017 г., протокол № 2.

Председатель _____ / С.И. Лукьянов/

Рабочая программа составлена: доцентом каф. ВтиП

_____ / М.В. Зарецкам/
(подпись) *(И.О. Фамилия)*

Рецензент: начальник отдела инновационных разработок ЗАО «КонсОмСКС», канд. техн. наук

_____ / А.Н. Панов/

Лист актуализации рабочей программы

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2017-2018 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 26 09 2017 г. № 2
Зав. кафедрой _____ О.С. Логунова

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2018 - 2019 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 5 09 2018 г. № 1
Зав. кафедрой _____ О.С. Логунова

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2019 - 2020 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 19 02 2020 г. № 5
Зав. кафедрой _____ О.С. Логунова

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2020 - 2021 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 19 08 2020 г. № 5
Зав. кафедрой _____ О.С. Логунова

1 Цели освоения дисциплины (модуля)

Целями освоения дисциплины «Основы машинного обучения» являются:

- формирование у студентов понимания современной методологии машинного обучения;
- формирование у студентов умения применять современные нечеткологические и нейросетевые методы;
- формирование у студентов навыков осознанного выбора и эффективного применения современных программных средств.
- Для достижения поставленных целей в курсе «Основы машинного обучения» решаются задачи:
- изучение методологических основ машинного обучения;
- изучение алгоритмических основ машинного обучения;
- освоение современного программного обеспечения, реализующего методы машинного обучения.

2 Место дисциплины (модуля) в структуре образовательной программы подготовки бакалавра (магистра, специалиста)

Дисциплина «Основы машинного обучения» входит в вариативную часть блока 1 образовательной программы.

Для изучения дисциплины необходимы знания (умения, владения), сформированные в результате изучения следующих дисциплин:

- философии (базовая часть блока 1 образовательной программы). Знания, полученные при изучении данной дисциплины, позволят обучающимся освоить основы эпистемологии, необходимые для понимания методологии информационного поиска;
- математики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, позволят обучающимся освоить математический аппарат информационного поиска;
- информатики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, являются основой для освоения средств обработки информации в соответствии с методологией информационного поиска;
- прикладного программирования (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, являются основой для освоения методологии разработки программ в области информационного поиска.

Знания (умения, владения), полученные при изучении данной дисциплины будут необходимы для выполнения выпускной квалификационной работы.

3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) и планируемые результаты обучения

В результате освоения дисциплины (модуля) «Методы нейрокомпьютерного моделирования» обучающийся должен обладать следующими компетенциями:

Структурный элемент компетенции	Планируемые результаты обучения
ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.	
Знать	– основные принципы анализа информации; основы концепций Data Mining, Text Mining, WEB Scraping;

Структурный элемент компетенции	Планируемые результаты обучения
	– современные методы мягких вычислений, применяемых при интеллектуальном анализе информации; – современные средства глубокого обучения и методологию их применения.
Уметь	– выбирать приемлемые алгоритмы и применять их для решения конкретных задач обработки информации; – самостоятельно конструировать алгоритмы обработки информации в нестандартных ситуациях; – конструировать сложные мультипарадигменные алгоритмы для анализа разнородной и неструктурированной информации.
Владеть	– навыками применения программных средств анализа информации; – навыками настройки сложных систем анализа информации; – навыками разработки программных средств анализа информации.
ОПК-5 Обладает способностью решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.	
Знать	– основные принципы машинного обучения; – современные интеллектуальные технологии машинного обучения; – методологию совершенствования систем машинного обучения.
Уметь	– выбирать концепцию построения модели интеллектуальной системы анализа информации, соответствующую поставленной прикладной задаче; – выбирать алгоритмы верификации функционирования моделей анализа информации.
Владеть	– навыками применения программного обеспечения интеллектуальных систем для разработки средств анализа информации; – навыками осуществления настройки и верификации программного обеспечения интеллектуальных систем для разработки и функционирования интеллектуальных моделей анализа информации; – навыками осуществления модификации программного обеспечения интеллектуальных систем для разработки и функционирования интеллектуальных моделей анализа информации.
ОК-6 Обладает способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия.	
Знать	– основные принципы организации и функционирования микросоциума; – методы предотвращения и разрешения конфликтов; – методологию социального проектирования.
Уметь	– анализировать состояние коллектива; – находить способы решения конкретных конфликтных ситуаций; – проектировать развитие коллектива в желательном направлении.
Владеть	– навыками межкультурной коммуникации; – навыками выстраивания системы стабильного развития в коллективе; – навыками оптимального целеполагания для каждого сотрудника и всего коллектива

4 Структура и содержание дисциплины (модуля)

Общая трудоемкость дисциплины составляет 4 зачетные единицы 144 акад. часа, в том числе:

- контактная работа – 55 акад. часов:
  - аудиторная – 54 акад. часа;
  - внеаудиторная – 1 акад. час
- самостоятельная работа – 89 акад. часов.

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
1. Раздел 1. Введение в методы машинного обучения.	7							
1.1. Тема. Основы работы с текстовой информацией.	7	2	4		8	Самостоятельное изучение учебной и научной литературы.	Беседа – обсуждение. Устный опрос.	*ОК-6 - зув* *ОПК-5 – зув* *ПК-2 – зув*
1.2. Тема. Анализ информации на основе персептроных моделей	7	2	4		8	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ОК-6 - зув* *ОПК-5 – зув* *ПК-2 – зув*
Итого по разделу	**7**	**4**	**8**		**16**		**Проверка индивидуальных заданий**	
2. Раздел. Анализ текстовой информации.	7							
2.1. Тема. Анализ «сырого» текста. Токенизация. Работа с корпусами текстов Программные средства для работы с	7	2	4		10	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ОК-6 - зув* *ОПК-5 – зув*

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
текстами.						занятию. Выполнение лабораторной работы.		*ПК-2 – зув*
2.2 Тема. Аннотирование текстов. Задачи анализа текстовой информации. Специфика работы с информацией, размещенной в сети Интернет.	7	2	4		10	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ОК-6 - зув* *ОПК-5 – зув* *ПК-2 – зув*
Итого по разделу	**7**	**4**	**8**		**20**		**Проверка индивидуальных заданий**	
3. Раздел. Нейросетевые методы извлечения информации. Нечеткологические методы извлечения информации.	7							
3.1. Тема. Ассоциативные нейронные сети в задачах извлечения информации.	7	2	4		12	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ОК-6 - зув* *ОПК-5 – зув* *ПК-2 – зув*
3.2. Тема. Нечеткологические методы в задачах извлечения информации.	7	2	4/2И		14	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной ра-	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ОК-6 - зув* *ОПК-5 – зув* *ПК-2 – зув*

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
						боты.		
Итого по разделу	**7**	**4**	**8/2И**		**26**		**Проверка индивидуальных заданий**	
4. Раздел. Методы глубокого обучения.	7							
4.1. Тема. Современные программные средства глубокого обучения.	7	4	12/12И		27	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ОК-6 - зув* *ОПК-5 – зув* *ПК-2 – зув*
Итого по разделу		**6**	**12/12И**		**27**		**Проверка индивидуальных заданий**	
Итого за семестр		**18**	**36/14И**		**89**		**Зачет**	
Итого по дисциплине		**18**	**36/14И**		**89**			

5 Образовательные и информационные технологии

1. **Традиционные образовательные технологии** ориентируются на организацию образовательного процесса, предполагающую прямую трансляцию знаний от преподавателя к студенту (преимущественно на основе объяснительно-иллюстративных методов обучения). Учебная деятельность студента носит в таких условиях, как правило, репродуктивный характер.

Формы учебных занятий с использованием традиционных технологий:

Информационная лекция – последовательное изложение материала в дисциплинарной логике, осуществляемое преимущественно вербальными средствами (монолог преподавателя).

Семинар – беседа преподавателя и студентов, обсуждение заранее подготовленных сообщений по каждому вопросу плана занятия с единым для всех перечнем рекомендуемой обязательной и дополнительной литературы.

Практическое занятие, посвященное освоению конкретных умений и навыков по предложенному алгоритму.

Лабораторная работа – организация учебной работы с реальными материальными и информационными объектами, экспериментальная работа с аналоговыми моделями реальных объектов.

2. **Технологии проблемного обучения** – организация образовательного процесса, которая предполагает постановку проблемных вопросов, создание учебных проблемных ситуаций для стимулирования активной познавательной деятельности студентов.

3. **Интерактивные технологии** – организация образовательного процесса, которая предполагает активное и нелинейное взаимодействие всех участников, достижение на этой основе личностно значимого для них образовательного результата. Наряду со специализированными технологиями такого рода принцип интерактивности прослеживается в большинстве современных образовательных технологий. Интерактивность подразумевает субъект - субъектные отношения в ходе образовательного процесса и, как следствие, формирование саморазвивающейся информационно-ресурсной среды.

Формы учебных занятий с использованием специализированных интерактивных технологий:

Лекция «обратной связи» – лекция–провокация (изложение материала с заранее запланированными ошибками), лекция-беседа, лекция-дискуссия, лекция–пресс-конференция.

4. **Информационно-коммуникационные образовательные технологии** – организация образовательного процесса, основанная на применении специализированных программных сред и технических средств работы с информацией.

6 Учебно-методическое обеспечение самостоятельной работы обучающихся

Задание к лабораторной работе по теме:

Основы работы с текстовой информацией.

Дана программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием.

1.

```
def myStr():
    """This is the program myStr"""
    s = 'Строка'
    st = type(s)
    print(s)
    print(st)
    sc1 = s.encode(encoding = "cp1251")
    print(sc1)
```

```
    sc2 = s.encode(encoding = "utf-8")
    print(sc2)
```

2.

```
def myStr():
    """This is the program myStr"""
    s = bytes("стр str", "cp1251")
    print(s[0],s[5],s[0:3],s[4:7])
    print(s)
```

3.

```
def myStr():
    """This is the program myStr"""
    s1 = "строка"
    l1 = len(s1)
    print(s1, l1)
    s2 = bytes("строка", "cp1251")
    l2 = len(s2)
    print(s2, l2)
    s3 = bytes("строка", "utf-8")
    l3 = len(s3)
    print(s3, l3)
```

4.

```
def myStr():
    """This is the program Str_04"""
    s = bytearray("str", "utf-8")
    print(s)
    s[0]=49
    print(s)
    s.append(55)
    print(s)
```

5.

```
def myStr():
    """This is the program myStr"""
    s1 = str(b"\xf1\xf2\xf0\xee\xea\xe0")
    print(s1)
    s2 = str(b"\xf1\xf2\xf0\xee\xea\xe0", "cp1252")
    print(s2)
```

6.

```
def myStr():
    """This is the program myStr"""
    obj1 = bytes("строка1", "utf-8")
    obj2 = bytearray("строка2", "utf-8")
    print(obj1)
    print(obj2)
    s1 = str(obj1, "utf-8")
    s2 = str(obj2, "utf-8")
    print(s1)
    print(s2)
    st1 = str(obj1,"ascii","ignore")
```

```
    st2 = str(obj1,"ascii","replace")
    print(st1)
    print(st2)
```

7.

```
def myStr():
    """This is the program myStr"""
    obj1 = bytes("строка1", "utf-8")
    obj2 = bytearray("строка2", "utf-8")
    st1 = str(obj1,"ascii","ignore")
    st2 = str(obj1,"ascii","replace")
    print(st1)
    print(st2)
    st3 = str(obj1,"cp1251","ignore")
    st4 = str(obj1,"cp1251","replace")
    print(st3)
    print(st4)
    st5 = str(obj1,"utf-8","ignore")
    st6 = str(obj1,"utf-8","replace")
    st7 = str(obj1,"utf-8","strict")
    print(st5)
    print(st6)
    print(st7)
```

8.

```
def myStr():
    """This is the program myStr"""
    s1 = 'строка1\nстрока2'
    print(s1)
    s2 = "строка1\nстрока2"
    print(s2)
    s3 = '''строка1\nстрока2'''
    print(s3)
    s4 = """строка1\nстрока2"""
    print(s4)
    s5=r"""строка1\nстрока2"""
    print(s5)
```

9.

```
def myStr():
    """This is the program myStr"""
    s1 = '\74'
    print(s1)
    s2 = '\x6a'
    print(s2)
    s3 = '\u043a'
    print(s3)
```

10.

```
def myStr():
    """This is the program myStr"""
    s = 'Python'
    print(s)
```

```
s1 = s[::-1]
print(s1)
s2 = 'J'+s[1::]
print(s2)
p = 'y' in s
print(p)
```

Задание к лабораторной работе по теме:

Анализ информации на основе персептроных моделей.

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием. Путь к файлам с исходными данными пользователь задает самостоятельно.

```
import numpy as np
from numpy.random import seed
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

class Perceptron(object):
    def __init__(self, eta=0.01, n_iter=10):
        self.eta=eta
        self.n_iter=n_iter

    def fit(self, X,y):
        self.w_ = np.zeros(1 +X.shape[1])
        self.errors_ = []

        for _ in range(self.n_iter):
            errors = 0
            for Xi, target in zip(X,y):
                update = self.eta * (target-self.predict(Xi))
                self.w_[1:] += update*Xi
                self.w_[0] += update
                errors += int(update != 0.0)
            self.errors_.append(errors)
        return self
    def net_input(self, X):
        return np.dot(X,self.w_[1:]) + self.w_[0]

    def predict(self, X):
        return np.where(self.net_input(X) >=0,1,-1)
```

1.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
```

```
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.1,n_iter=10)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

2.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.05,n_iter=50)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

3.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.09,n_iter=170)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

4.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
```

```
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.2,n_iter=7)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

5.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.09,n_iter=15)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

6.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='magenta', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.08,n_iter=10)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

7.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
```

```
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.05,n_iter=100)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

8.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.06,n_iter=1000)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

9.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.07,n_iter=110)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

10.

```
fname=r'c:/Users/user/Anaconda3/Lib/site-packages/pandas/tests/data/iris.csv'
df = pd.read_csv(fname, header=None)
print(df.head())
y = df.iloc[1:100,4].values
y = np.where(y == 'Iris-setosa', -1, 1)
for i in range(X.shape[0]):
    X[i,0],X[i,1]=float(X[i,0]),float(X[i,1])
X = df.iloc[1:100,[0,2]].values
plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')
```

```
plt.scatter(X[50:100], X[50:100], color='blue', marker='x', label='versicolor')
plt.show()
ppn = Perceptron(eta=0.03,n_iter=190)
ppn.fit(X,y)
print(ppn.errors_)
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.show()
```

Задание к лабораторной работе по теме:

Анализ «сырого» текста. Токенизация. Работа с корпусами текстов Программные средства для работы с текстами.

Рассмотреть предложенный пример на языке Python, предназначенный для работы с сырым текстом и текстовыми корпусами.

1.
```
from nltk.corpus import gutenberg
from nltk import FreqDist
def Ling_01():
    print(gutenberg.fileids())

def Ling_02():
    fd = FreqDist()
    for word in gutenberg.words('austen-persuasion.txt'):
        fd.inc(word)
    print(fd.N())
    print(fd.B())
```

2.
```
from nltk.corpus import gutenberg
from nltk import FreqDist
def Ling_01():
    print(gutenberg.fileids())

def Ling_02():
    fd = FreqDist()
    for word in gutenberg.words('austen-persuasion.txt'):
        fd[word]+=1
    print(fd.N())
    print(fd.B())
```

3.
```
from nltk.book import *
def Sample_01():
    print(text1)
    print(text2)
    print(text3)

def Sample_02():
    C1 = text1.concordance('monstrous')
    C2 = text3.concordance('God')
    print(C1)
    print(C2)
```

4.
```
from nltk.book import *
def Sample_02():
    print('For monstrous')
    text1.concordance('monstrous')
    print('For great')
    text2.concordance('great')
    print('For God')
    text3.concordance('God')
```

5.
```
from nltk.book import *
def Sample_03():
    text1.similar('monstrous')
    text2.similar('little')
    text3.similar('God')
```

6.
```
from nltk.book import *
def Sample_04():
    text1.common_contexts(['monstrous', 'very'])
    text2.common_contexts(['little', 'great'])
    text3.common_contexts(['God', 'devil'])
```

7.
```
import nltk
def corp_01():
    ff = nltk.corpus.gutenberg.fileids()
    print(ff)
def corp_02():
    emma = nltk.corpus.gutenberg.words('austen-emma.txt')
    ll = len(emma)
    print(qq)
```

8.
```
import nltk
def corp_01():
    ff = nltk.corpus.gutenberg.fileids()
    print(ff)
def corp_02():
    emma = nltk.corpus.gutenberg.words('austen-emma.txt')
    ll = len(emma)
    print(ll)
def corp_03():
    emma = nltk.Text(nltk.corpus.gutenberg.words('austen-emma.txt'))
    emma.concordance('surprise')
```

9.
```
def WEB_02():
    url = "http://gutenberg.spiegel.de/buch/belagerung-von-mainz-3641/1"
#    url = "http://gutenberg.spiegel.de/buch/achilleis-7287/1"
    html=request.urlopen(url).read().decode('utf8')
```

```
    print(html[:60])
    raw=BeautifulSoup(html, 'html.parser').get_text()
    print(raw)
    tokenizer = TreebankWordTokenizer()
    tokens = tokenizer.tokenize(raw)
    tokens = tokens[110:390]
    print(tokens)
    text = nltk.Text(tokens)
    print(text)
    conc = text.concordance('gene')
    print(conc)
```

10.

```
def Disk_01():
    f = open('C:/Python_Prog/Deutsch/Goethe_02.txt','r',encoding='utf-8')
    raw = f.read()
    print(raw)
    f.close()
    german_tokenizer = nltk.data.load(
    'tokenizers/punkt/german.pickle')
    tokens = german_tokenizer.tokenize(raw)
    tokens = tokens[:500]
    print(tokens)
    text = nltk.Text(tokens)
    print(text)
    conc = text.concordance('das',lines=100)
    print(conc)
```

Задание к лабораторной работе по теме:

Аннотирование текстов. Задачи анализа текстовой информации. Специфика работы с информацией, размещенной в сети Интернет.

Рассмотреть предложенный пример на языке Python, предназначенный для работы с текстом, размещенным в Интернет. Всем программам предшествует вызов модулей:

```
from urllib.request import urlopen
from urllib.error import HTTPError
from bs4 import BeautifulSoup
```

1.

```
def scr():
    path=r"http://pythonscraping.com/pages/page1.html"
    html=urlopen(path)
    print(html.read())
```

2.

```
def scr():
    path=r"http://pythonscraping.com/pages/page1.html"
    html=urlopen(path)
    bsObj=BeautifulSoup(html.read())
    print(bsObj.h1)
```

3.

```
def scr():
    path=r"http://pythonscraping.com/pages/page111.html"
```

```
    try:
        html=urlopen(path)
    except HTTPError as e:
        print(e)
    else:
        if html is None:
            print("URL is not found")
        else:
            bsObj=BeautifulSoup(html.read())
```

4.

```
def getTitle(url):
    try:
        html=urlopen(url)
    except HTTPError as e:
        return None
    try:
      bsObj=BeautifulSoup(html.read())
      title=bsObj.body.h1
    except AttributeError as e:
        return None
    return title
```

5.

```
def scr():
    url=r"http://www.pythonscraping.com/pages/page11.html"
    title=getTitle(url)
    if title==None:
        print("Title could not be found")
    else:
        print(title)
```

6.

```
def wscr():
    url=r"http://www.pythonscraping.com/pages/warandpeace.html"
    html=urlopen(url)
    bsObj=BeautifulSoup(html)
    nameList=bsObj.findAll("span",{"class":"green"})
    for name in nameList:
        print(name.get_text())
```

7.

```
def wscr():
    url=r"http://www.pythonscraping.com/pages/warandpeace.html"
    html=urlopen(url)
    bsObj=BeautifulSoup(html)
    nameList=bsObj.findAll(text="the prince")
    print(len(nameList))
```

8.

```
def wscr_03():
    url=r"http://www.pythonscraping.com/pages/warandpeace.html"
    html=urlopen(url)
```

```
bsObj=BeautifulSoup(html)
allText=bsObj.findAll(id="text")
print(allText[0].get_text())
```

9.

```
def webscr_01():
    url=r"http://en.wikipedia.org/wiki/Kevin_Bacon"
    html=urlopen(url)
    bsObj=BeautifulSoup(html)
    for link in bsObj.findAll("a"):
        if 'href' in link.attrs:
            print(link.attrs['href'])
```

10.

```
def wscr():
    url=r"http://www.pythonscraping.com/pages/warandpeace.html"
    html=urlopen(url)
    bsObj=BeautifulSoup(html)
    nameList=bsObj.findAll("span",{"class":"red"})
    for name in nameList:
        print(name.get_text())
```

Задание к лабораторной работе по теме:
Ассоциативные нейронные сети в задачах извлечения информации.

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием.

```
class HammingNeuron:

    def __init__(self, weights, next_neuron=None):
        self.weights = list()
        self.inputs = list()
        self.next_neuron = None
        for w in weights:
            self.weights.append(w)
            self.inputs.append(0)
            self.next_neuron = next_neuron

    def change_weight(self, ind_of_weight, new_value):
        self.weights[ind_of_weight] = new_value

    def set_input(self, ind_of_input, value):
        self.inputs[ind_of_input] = value

    def set_next_neuron(self, next_neuron):
        self.next_neuron = next_neuron

    def count_output(self):
        res = 1/2 + sum(self.inputs[i] * self.weights[i] for i in range(0, len(self.weights)))/(2 *
len(self.weights))
        return res

    def get_output(self):
```

```
            self.next_neuron.set_value(self.count_output())


        class MaxNetNeuron:

            def __init__(self, index, weights, next_neuron):
                self.value = 0
                self.reinitial_value = 0
                self.inputs = list()
                self.weights = list()
                self.layer_neurons = list()
                self.index = index
                self.next_neuron = next_neuron
                for w in weights:
                    self.weights.append(w)
                    self.inputs.append(None)

            def set_layer_neurons(self, layer_neurons):
                for n in layer_neurons:
                    self.layer_neurons.append(n)

            def set_value(self, value):
                self.value = value
                self.reinitial_value = value

            def set_only_current_value(self, value):
                self.value = value

            def set_input(self, ind_of_neuron, value):
                self.inputs[ind_of_neuron] = value

            def count_output(self):
                # if first time
                if self.inputs[self.index] is None:
                    return self.value
                # if not first time
                else:
                    return self.inputs[self.index] - \
                        sum(self.inputs[i] * self.weights[i] for i in range(0, len(self.weights)) if i !=
self.index)

            def recount_value(self):
                self.value = self.count_output()

            def get_output_inside_layer(self):
                for n in self.layer_neurons:
                    n.set_input(self.index, self.value)

            def get_output(self):
                self.next_neuron.set_value(self.value)

            def reinitialize_neuron(self):
                self.value = self.reinitial_value
```

```
        for i in range(0, len(self.inputs)):
            self.inputs[i] = None


class ThresholdNeuron:

    def __init__(self, index, next_neurons):
        self.value = None
        self.next_neurons = list()
        self.index = index
        for n in next_neurons:
            self.next_neurons.append(n)

    def set_value(self, value):
        self.value = value

    def count_output(self):
        if self.value > 0:
            return 1
        else:
            return 0

    def get_output(self):
        for n in self.next_neurons:
            n.set_input(self.index, self.count_output())


class OutputNeuron:

    def __init__(self, weights):
        self.weights = list()
        self.inputs = list()
        for w in weights:
            self.weights.append(w)
            self.inputs.append(0)

    def set_input(self, index, value):
        self.inputs[index] = value

    def get_output(self):
        return sum(self.weights[i] * self.inputs[i] for i in range(0, len(self.weights)))


class HammingLayer:

    def __init__(self, weights, next_neurons):
        self.neurons = list()
        for i in range(0, len(weights)):
            new_neuron = HammingNeuron(weights[i], next_neurons[i])
            self.neurons.append(new_neuron)

    def run(self, inputs):
        for n in self.neurons:
```

```
            for i in range(0, len(inputs)):
                n.set_input(i, inputs[i])
        for n in self.neurons:
            n.get_output()


class MaxNetLayer:

    def __init__(self, next_neurons):
        self.neurons = list()
        k = len(next_neurons)
        for i in range(0, k):
            weights = [random.random() * 1/(k - 1) for j in range(0, i)] + [1] +\
                    [random.random() * 1/(k  - 1) for j in range(i + 1, k)]
            new_neuron = MaxNetNeuron(i, weights, next_neurons[i])
            self.neurons.append(new_neuron)
        for n in self.neurons:
            n.set_layer_neurons(self.neurons)

    def run(self):
        for n in self.neurons:
            n.get_output_inside_layer()
        for n in self.neurons:
            n.recount_value()
        for n in self.neurons:
            n.get_output()

    def reinitialize_layer(self, num_of_not_null_neuron, eps):
        self.neurons[num_of_not_null_neuron].reinitial_value -= eps
        for n in self.neurons:
            n.reinitialize_neuron()


class ThresholdLayer:

    def __init__(self, count, next_neurons):
        self.neurons = list()
        for i in range(0, count):
            new_neuron = ThresholdNeuron(i, next_neurons)
            self.neurons.append(new_neuron)

    def run(self):
        for n in self.neurons:
            n.get_output()

    def get_first_not_null_element(self):
        for n in self.neurons:
            if n.count_output() == 1:
                return self.neurons.index(n)


class OutputLayer:
```

```
        def __init__(self, weights):
            self.neurons = list()
            for i in range(0, len(weights[0])):
                self.neurons.append(OutputNeuron([weights[j][i] for j in range(0, len(weights))]))

        def get_result(self):
            l = []
            for n in self.neurons:
                l.append(n.get_output())
            return l


    class HammingNetwork:
        """
        Initial arguments:
            learning_examples - list of learning examples
            eps - maximal distance between winners
            max_count_of_outputs - maximal count of winners
        """

        def __init__(self, learning_examples, eps, max_count_of_outputs):
            self.max_count_of_outputs = max_count_of_outputs
            self.eps = eps
            self.output_layer = OutputLayer(learning_examples)
            self.threshold_layer = ThresholdLayer(len(learning_examples),
self.output_layer.neurons)
            self.max_net_layer = MaxNetLayer(self.threshold_layer.neurons)
            self.hamming_layer = HammingLayer(learning_examples,
self.max_net_layer.neurons)

        def classification(self, example_inputs):
            res = []
            self.hamming_layer.run(example_inputs)
            first_time = True
            while(True):
                while(True):
                    self.max_net_layer.run()
                    if sum(n.count_output() for n in self.threshold_layer.neurons) == 1:
                        break
                    for n in self.max_net_layer.neurons:
                        if n.next_neuron.count_output() == 0:
                            n.set_only_current_value(0)
                self.threshold_layer.run()
                res.append(self.output_layer.get_result())
                if first_time:
                    first_time = False

self.max_net_layer.reinitialize_layer(self.threshold_layer.get_first_not_null_element(), self.eps)
                    continue
                else:
                    if res[len(res) - 1] in res[0:len(res) - 1] or len(res) > self.max_count_of_outputs:
                        res = res[0:len(res) - 1]
                        return res
```

```
                else:
self.max_net_layer.reinitialize_layer(self.threshold_layer.get_first_not_null_element(), self.eps)
                    continue
```

1.

```
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, -1, 1, 1,
-1, 1],
                       '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                       '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                       '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
        print('----------------------------------------')
```

2.

```
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                       '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                       '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                       '9': [1, 1, 1, 1, -1, -1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.4
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
        print('----------------------------------------')
```

3.

```
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                       '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
```

```
                    '6': [1, 1, -1, 1, 1, 1, 1], '7': [1, 1, 1, -1, 1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                    '9': [1, 1, 1, 1, -1, 1, 1]}
        my_learning_examples =[ '0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
        my_validation_examples = ['6', '7', '8']
        my_eps = 0.3
        max_count_of_output = 3
        my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
        for ex in my_validation_examples:
            print('example')
            print(str(ex))
            for ans in my_hamming_network.classification(dict_of_numbers[ex]):
                print('answer')
                print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
            print('----------------------------------------')
```

4.

```
    if __name__ == '__main__':
        dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                    '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                    '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                    '9': [1, 1, 1, 1, -1, 1, 1]}
        my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
        my_validation_examples = ['6', '7', '8']
        my_eps = 0.3
        max_count_of_output = 3
        my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
        for ex in my_validation_examples:
            print('example')
            print(str(ex))
            for ans in my_hamming_network.classification(dict_of_numbers[ex]):
                print('answer')
                print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
            print('----------------------------------------')
```

5.

```
    if __name__ == '__main__':
        dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                    '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                    '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                    '9': [1, 1, 1, 1, -1, 1, 1]}
        my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
        my_validation_examples = ['6', '7', '8']
        my_eps = 0.3
        max_count_of_output = 3
        my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
        for ex in my_validation_examples:
            print('example')
            print(str(ex))
```

```
            for ans in my_hamming_network.classification(dict_of_numbers[ex]):
                print('answer')
                print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
            print('----------------------------------------')
```

6.

```
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, 1, 1,1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1, -1, 1],
                       '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                       '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                       '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
        print('----------------------------------------')
```

7.

```
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1, -1, 1],
                       '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                       '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                       '9': [1, 1, 1, 1, -1, 1, 1]}
    my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    my_validation_examples = ['6', '7', '8']
    my_eps = 0.3
    max_count_of_output = 3
    my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in my_learning_examples], my_eps, max_count_of_output)
    for ex in my_validation_examples:
        print('example')
        print(str(ex))
        for ans in my_hamming_network.classification(dict_of_numbers[ex]):
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
        print('----------------------------------------')
```

8.

```
if __name__ == '__main__':
    dict_of_numbers = {'0': [1, 1, 1, -1, 1,1,1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1, -1, 1],
                       '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                       '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
```

```
                    '9': [1, 1, 1, 1, -1, 1, 1]}
        my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
        my_validation_examples = ['6', '7', '8']
        my_eps = 0.3
        max_count_of_output = 3
        my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
        for ex in my_validation_examples:
            print('example')
            print(str(ex))
            for ans in my_hamming_network.classification(dict_of_numbers[ex]):
                print('answer')
                print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
            print('----------------------------------------')
```

9.

```
    if __name__ == '__main__':
        dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                    '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                    '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                    '9': [1, 1, 1, 1, -1, 1, 1]}
        my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
        my_validation_examples = ['6', '7', '8']
        my_eps = 0.3
        max_count_of_output = 3
        my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
        for ex in my_validation_examples:
            print('example')
            print(str(ex))
            for ans in my_hamming_network.classification(dict_of_numbers[ex]):
                print('answer')
                print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
            print('----------------------------------------')
```

10.

```
    if __name__ == '__main__':
        dict_of_numbers = {'0': [1, 1, 1, -1, 1, 1, 1], '1': [-1, -1, 1, -1, -1, 1, -1], '2': [-1, 1, 1, 1, 1,
-1, 1],
                    '3': [-1, 1, 1, 1, -1, 1, 1], '4': [1, -1, 1, 1, -1, 1, -1], '5': [1, 1, -1, 1, -1, 1, 1],
                    '6': [1, 1, -1, 1, 1, 1, 1], '7': [-1, 1, 1, -1, -1, 1, -1], '8': [1, 1, 1, 1, 1, 1, 1],
                    '9': [1, 1, 1, 1, -1, 1, 1]}
        my_learning_examples = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
        my_validation_examples = ['6', '7', '8']
        my_eps = 0.3
        max_count_of_output = 3
        my_hamming_network = HammingNetwork([dict_of_numbers[k] for k in
my_learning_examples], my_eps, max_count_of_output)
        for ex in my_validation_examples:
            print('example')
            print(str(ex))
            for ans in my_hamming_network.classification(dict_of_numbers[ex]):
```

```
            print('answer')
            print([key for key, val in list(dict_of_numbers.items()) if val == ans][0])
        print('----------------------------------------')
```

Задание к лабораторной работе по теме:

Нечеткологические методы в задачах извлечения информации.

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием.

1.

```
import fuzzywuzzy
from fuzzywuzzy import fuzz,process
def stFuzz_01():
    "This is a simple program which uses fuzzy technologies"
    print('Fuzzy!')
    s1,s2="This is a test","This is a test!"
    fuzz_ratio,fuzz_part_ratio = fuzz.ratio(s1,s2),fuzz.partial_ratio(s1,s2)
    print(fuzz_ratio,fuzz_part_ratio,sep='__&__')
```

2.

```
import fuzzywuzzy
from fuzzywuzzy import fuzz,process
def stFuzz_02():
    "This is a simple program which uses fuzzy technologies"
    print('Fuzzy!')
    s1,s2="A cat eats a rat die nächtlichen!","A rat eats a cat die nächtlichen!"
    fuzz_ratio,fuzz_part_ratio = fuzz.ratio(s1,s2),fuzz.partial_ratio(s1,s2)
    print(fuzz_ratio,fuzz_part_ratio,sep='__&__')
```

3.

```
import fuzzywuzzy
from fuzzywuzzy import fuzz,process
def stFuzz_03():
    "This is a simple program which uses fuzzy technologies"
    key="Testing FuzzyWuzzy"
    choices=['Testing Fuzzy Wuzzy','Testing Wuzzy','FuzzyWuzzy Testing',
             'Testing WuzzyFuzzy','Testing fuzzy wuzzy','fuzzy wuzzy test',
             'Testing']
    proc_extr=process.extract(key,choices,scorer=fuzz.ratio)
    print(proc_extr)
    proc_extr=process.extract(key,choices,scorer=fuzz.ratio,limit=7)
    print(proc_extr)
    proc_extr=process.extractOne(key,choices,scorer=fuzz.ratio)
    print(proc_extr)
```

4.

```
import fuzzywuzzy
from fuzzywuzzy import fuzz,process
def stFuzz_04():
    """This is a simple program which uses fuzzy technologies"""
    key="Kovaleva"
    choices=['Kovaliova','Kovalyova','Kowaljowa','Kowalowa','Kowaliowa']
    proc_extr=process.extract(key,choices,scorer=fuzz.ratio)
```

```
    print(type(proc_extr),proc_extr)
    q=proc_extr[0]
    print(type(q),q)
    r,s=q[0],q[1]
    print(type(r),r,type(s),s)
```

5.

```
import fuzzywuzzy
from fuzzywuzzy import fuzz,process
def stFuzz_05():
    """This is a simple program which uses fuzzy technologies"""
    key="Kovaleva"
    choices=['Kovaliova','Kovalyova','Kowaljowa','Kowalowa','Kowaliowa']
    proc_extr1=process.extract(key,choices,scorer=fuzz.ratio)
    proc_extr2=process.extract(key,choices,scorer=fuzz.partial_ratio)
    fuz_set1,fuz_set2 = {key:1},{key:1}
    for item in proc_extr1:
       q,r=item[0],item[1]*0.01
       fuz_set1[q]=r
    print(fuz_set1)
    for item in proc_extr2:
       q,r=item[0],item[1]*0.01
       fuz_set2[q]=r
    print(fuz_set2)
```

6.

```
import fuzzywuzzy
from fuzzywuzzy import fuzz,process
def stFuzz_06():
    """This is a simple program which uses fuzzy technologies"""
    key="Cats eat mice"
    choices=['Mice eat cats','mice eats cats',
    'Cats Eat Mice ']
    proc_extr1=process.extract(key,choices,scorer=fuzz.token_sort_ratio)
    proc_extr2=process.extract(key,choices,scorer=fuzz.token_set_ratio)
    fuz_set1,fuz_set2 = {key:1},{key:1}
    for item in proc_extr1:
       q,r=item[0],item[1]*0.01
       fuz_set1[q]=r
    print(fuz_set1)
    for item in proc_extr2:
       q,r=item[0],item[1]*0.01
       fuz_set2[q]=r
```

7.

```
from difflib import SequenceMatcher
from fuzzywuzzy import fuzz
from stFuzzy_4 import levenshtein

def sim_1():
   s1,s2="NEW YORK METS","NEW YORK MEETS"
   s3,s4="A little cat!","A little cat."
   m1,m2=SequenceMatcher(None,s1,s2),SequenceMatcher(None,s3,s4)
```

```
print(m1.ratio(),m2.ratio())
```

8.

```
from difflib import SequenceMatcher
from fuzzywuzzy import fuzz
from stFuzzy_4 import levenshtein
def sim_2():
   s1,s2="GOOD EVENING!","Good evening!"
   m1,m2=SequenceMatcher(None,s1,s2),fuzz.ratio(s1,s2)
   m3=fuzz.partial_ratio(s1,s2)
   print(m1.ratio(),m2,m3)
```

9.

```
from difflib import SequenceMatcher
from fuzzywuzzy import fuzz
from stFuzzy_4 import levenshtein
def sim_4():
   s1,s2="GOD","DOG"
   m1,m2=SequenceMatcher(None,s1,s2),fuzz.ratio(s1,s2)
   m3=fuzz.partial_ratio(s1,s2)
   print(m1.ratio(),m2,m3)
```

10.

```
from difflib import SequenceMatcher
from fuzzywuzzy import fuzz
from stFuzzy_4 import levenshtein
def sim_5():
   s1,s2="NEW YORK METS","NEW YORK MEETS"
   s3,s4="A little cat","A little cat!"
   lev1,lev2=levenshtein(s1,s2),levenshtein(s3,s4)
   print(lev1,lev2,lev3,lev4)
```

Задание к лабораторной работе по теме:

Современные программные средства глубокого обучения.

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить свое задание, воспользовавшись приведенными программами. Для выполнения задания потребуется существенная переработка программ.

Программа 1

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
dataset = np.loadtxt("data.txt", delimiter=",")
# Первые 8 столбцов в примере отвечают за фичи, последний же за класс, разбиваем
X = dataset[:,0:8]
Y = dataset[:,8]
model = Sequential()
model.add(Dense(8, activation='relu'))
sigmoid , реже softmax
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, Y, epochs=15, batch_size=10,  verbose=2)
predictions = model.predict(X)
```

Программа 2

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
```

```
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

1. Обучить модель keras распознаванию рукописных букв «А», «Б», «В».
2. Обучить модель keras распознаванию рукописных букв «Г», «Д», «Е».
3. Обучить модель keras распознаванию рукописных букв «Ж», «З», «И».
4. Обучить модель keras распознаванию рукописных букв «К», «Л», «М».
5. Обучить модель keras распознаванию рукописных букв «О», «П», «Р».
6. Обучить модель keras распознаванию рукописных букв «С», «Т», «У».
7. Обучить модель keras распознаванию рукописных букв «Ф», «Х», «Ц».
8. Обучить модель keras распознаванию рукописных букв «Ч», «Ш», «Щ».
9. Обучить модель keras распознаванию рукописных букв «Э», «Ю», «Я».
10. Обучить модель keras распознаванию рукописных букв «I», «L», «Q».

Индивидуальные задания к разделу 1.

Дана самонастраивающаяся программа на языке Python. Проанализировать текст программы. Выполнить обращение к ней в соответствии с заданием.

```
import numpy as np

def nonlin(x, deriv=False):
    if (deriv):
        return nonlin(x)*(1-nonlin(x))
    return 1/(1+np.exp(-x))

def train(X,y,n_iter):
    np.random.seed(1)
    syn0=2*np.random.random((3,1))-1
    for iter in range(n_iter):
        l0 = X
        l1 = nonlin(np.dot(l0,syn0))
        l1_error=y-l1
        l1_delta=l1_error*nonlin(l1,True)
        syn0 += np.dot(l0.T,l1_delta)
    return syn0
def query(syn,XX):
    res = nonlin(np.dot(XX,syn))
    return res

1.
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,1,1,1]]).T
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 1000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)

2.
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,1 ,2],[0,2,2],[2,0,2],[2,2,2]])
```

```
n_iter = 15000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

3.

```
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,4],[0,3,2],[2,0,2],[2,3,2]])
n_iter = 11000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

4.

```
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,2,1],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

5.

```
X = np.array([[1,0,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,3,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

6.

```
X = np.array([[0,0,1],[1,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,3,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

7.

```
X = np.array([[0,0,1],[0,1,1],[1,0,1],[0,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[3,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
```

```
ans=query(syn,XX)
print(ans)
```
8.
```
X = np.array([[1,0,1],[0,1,1],[1,0,0],[1,0,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

9.
```
X = np.array([[0,1,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,0,2],[0,3,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

10.
```
X = np.array([[0,1,1],[0,1,1],[1,0,1],[1,1,1]])
y = np.array([[0,0,1,1]]).T
XX = np.array([[0,1,2],[0,2,2],[2,0,2],[2,2,2]])
n_iter = 10000
syn=train(X,y,n_iter)
print(syn)
ans=query(syn,XX)
print(ans)
```

Индивидуальные задания к разделу 2.

Рассмотреть предложенный пример на языке Python.

1.
```
def Ling_01():
    from nltk.corpus import genesis
    mygerman=genesis.words('german.txt')
    lmygerman = len(mygerman)
    print(lmygerman)
    tmygerman=nltk.Text(mygerman)
    conc1=tmygerman.concordance('Gott')
    print('sim')
    sim1=tmygerman.similar('Gott')
    print('cont')
    cont1=tmygerman.common_contexts(['Gott', 'sprach'])
    alph=sorted(set(tmygerman))
```

2.
```
def Ling_02():
    from nltk.corpus import genesis
    mygerman=genesis.words('german.txt')
    lmygerman = len(mygerman)
```

```
    print(lmygerman)
    tmygerman=nltk.Text(mygerman)
    alph=sorted(set(tmygerman))
    #print(alph)
    ltmygerman=len(tmygerman)
    lalph=len(alph)
    print(ltmygerman, lalph)
    lexical_diversity=ltmygerman/lalph
    print(lexical_diversity)
    fdistg=FreqDist(tmygerman)
    fdistg.plot(10,cumulative=True)

3.
def Ling_03():
    from nltk.corpus import swadesh
    mygerman=swadesh.words('de')
    lmygerman = len(mygerman)
    print(lmygerman)
    tmygerman=nltk.Text(mygerman)
#    conc1=tmygerman.concordance('Himmel')
    sim1=tmygerman.similar('Himmel')

4.
def Ling_04():
    from nltk.corpus import udhr
    langs=['English-Latin1', 'German_Deutsch-Latin1',
                'Russian_Russky-UTF8']
    cfd=nltk.ConditionalFreqDist(
                (lang,len(word))
                for lang in langs
                for word in udhr.words(lang))
    cfd.plot(cumulative=True)

5.
import requests
from bs4 import BeautifulSoup
path=r'http://dataquestio.github.io/web-scraping-pages/simple.html'
    page = requests.get(path)
    print(page)
    page_sc,page_cont = page.status_code,page.content
    print(page_sc,page_cont,sep='__&__')
    soup = BeautifulSoup(page_cont,'html.parser')
    soup_pretty = soup.prettify()
    print(soup_pretty)
    soup_child=soup.children
    soup_child_list = list(soup_child)
    print(soup_child_list)

6.
def scrap_02():
    path=r'http://dataquestio.github.io/web-scraping-pages/simple.html'
    page = requests.get(path)
    print(page)
```

```
    page_cont = page.content
    soup = BeautifulSoup(page_cont,'html.parser')
    soup_find_all=soup.find_all('p')
    print(soup_find_all)
    text=soup_find_all[0].get_text()
    print(text)
```

7.

```
def scrap_03():
    path1=r'http://dataquestio.github.io/web-scraping-pages/'
    path2=r'ids_and_classes.html'
    path=path1+path2
    page=requests.get(path)
    soup=BeautifulSoup(page.content,'html.parser')
    print(soup)
    soup_find_all_class=soup.find_all('p',class_='outer-text')
    print(soup_find_all_class)
    soup_find_all_id=soup.find_all(id='first')
    print(soup_find_all_id)
    soup_select_div=soup.select('div p')
    print(soup_select_div)
```

8.

```
def scrap_04():
     path1=r'http://forecast.weather.gov/'
     path2=r'MapClick.php?lat=37.7772&lon=-122.4168'
     path=path1+path2
     page=requests.get(path)
     soup=BeautifulSoup(page.content,'html.parser')
     seven_day=soup.find(id='seven-day-forecast')
     forecast_items=seven_day.find_all(class_='tombstone-container')
#     print(forecast_items)
     tonight=forecast_items[0]
     tonight_pretty=tonight.prettify()
     print(tonight_pretty)
     period=tonight.find(class_='period-name').get_text()
     short_desc=tonight.find(class_='short-desc').get_text()
     temp=tonight.find(class_='temp').get_text()
     print(period,short_desc,temp,sep='__&__')
     img=tonight.find('img')
     desc=img['title']
     print(img)
```

9.

```
import nltk
import nltk.data
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.tokenize import TreebankWordTokenizer
def token_01():
    sent1 = "Goog day! We are dlad! It's good!"
    sent2 = "Good day. We are glad. It,s good."
    sent3 = "Good day: We are glad; It;s good"
```

```
    tsent1 = sent_tokenize(sent1)
    tsent2 = sent_tokenize(sent2)
    tsent3 = sent_tokenize(sent3)
    ls1,ls2,ls3 = len(tsent1),len(tsent2),len(tsent3)
    print(ls1,tsent1)
    print(ls2,tsent2)
    print(ls3,tsent3)
```

10.

```
    english_tokenizer = nltk.data.load(
    'tokenizers/punkt/english.pickle')
    sent1 = "Good day! We are glad! It's good!"
    sent2 = "Good day. We are glad. It,s good."
    sent3 = "Good day: We are glad; It's good."
    tsent1 = english_tokenizer.tokenize(sent1)
    tsent2 = english_tokenizer.tokenize(sent2)
    tsent3 = english_tokenizer.tokenize(sent3)
    ls1,ls2,ls3 = len(tsent1),len(tsent2),len(tsent3)
    print(ls1,tsent1)
    print(ls2,tsent2)
    print(ls3,tsent3)
```

Индивидуальные задания к разделу 3.

Рассмотреть предложенный пример на языке Python.

1.

```
def levenshtein(s1, s2):
    if len(s1) < len(s2):
        return levenshtein(s2, s1)

    if len(s2) == 0:
        return len(s1)

    previous_row = range(len(s2) + 1)
    for i, c1 in enumerate(s1):
        current_row = [i + 1]
        for j, c2 in enumerate(s2):
            insertions = previous_row[j + 1] + 1 # j+1 instead of j since previous_row and cur-
rent_row are one character longer
            deletions = current_row[j] + 1       # than s2
            substitutions = previous_row[j] + (c1 != c2)
            current_row.append(min(insertions, deletions, substitutions))
        previous_row = current_row

    return previous_row[-1]
```

2.

```
def stFuzz_11():
    """This is a simple program which uses fuzzy technologies"""
    key="Котов Василий"
    choices=['Котов Васіль','Котов Василь']
    proc_extr1=process.extract(key,choices,scorer=fuzz.token_sort_ratio)
    proc_extr2=process.extract(key,choices,scorer=fuzz.token_set_ratio)
```

```
    fuz_set1,fuz_set2 = {key:1},{key:1}
    for item in proc_extr1:
       q,r=item[0],item[1]*0.01
       fuz_set1[q]=r
    print(fuz_set1)
    for item in proc_extr2:
       q,r=item[0],item[1]*0.01
       fuz_set2[q]=r
    print(fuz_set2)
```

3.

```
def stFuzz_10():
    """This is a simple program which uses fuzzy technologies"""
    key="Alia Burkhanova"
    choices=['Aliya Burhanova',Alya Burkhanova']
    proc_extr1=process.extract(key,choices,scorer=fuzz.token_sort_ratio)
    proc_extr2=process.extract(key,choices,scorer=fuzz.token_set_ratio)
    fuz_set1,fuz_set2 = {key:1},{key:1}
    for item in proc_extr1:
       q,r=item[0],item[1]*0.01
       fuz_set1[q]=r
    print(fuz_set1)
    for item in proc_extr2:
       q,r=item[0],item[1]*0.01
       fuz_set2[q]=r
    print(fuz_set2)
```

4.

```
def stFuzz_08():
    """This is a simple program which uses fuzzy technologies"""
    key="Chopin"
    choices=['Chopen','Szopen','Shopen ','Schopen']
    proc_extr1=process.extract(key,choices,scorer=fuzz.ratio)
    proc_extr2=process.extract(key,choices,scorer=fuzz.partial_ratio)
    fuz_set1,fuz_set2 = {key:1},{key:1}
    for item in proc_extr1:
       q,r=item[0],item[1]*0.01
       fuz_set1[q]=r
    print(fuz_set1)
    for item in proc_extr2:
       q,r=item[0],item[1]*0.01
       fuz_set2[q]=r
    print(fuz_set2)
```

5.

```
import re

def findWord(word):
   path = r'C:/Python_Prog/Deutsch/Goethe_02.txt'
   f = open(path,'r',encoding='utf-8')
   raw = f.read().splitlines()
   f.close()
   text = ' '.join(raw)
```

```
    preP = r"[^.?!]*(?<=[.?\s!])"+word+"(?=[\s.?!])[^.?!]*[.?!]"
    p = re.compile(preP, re.I|re.M)
    match = p.findall(text)
    print(match)
```

6.

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import RegexpStemmer
from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer

def corr_01():
    w1,w2="cooking","cookery"
    w3,w4="ingleside","machinery"
    stemmer=PorterStemmer()
    sw1,sw2=stemmer.stem(w1),stemmer.stem(w2)
    print(sw1,sw2,sep='__&__')
    sw3,sw4=stemmer.stem(w3),stemmer.stem(w4)
    print(sw3,sw4,sep='__&__')
```

7.

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import RegexpStemmer
from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer
def corr_04():
    SnowballLangs=SnowballStemmer.languages
    print(SnowballLangs)
    german_stemmer = SnowballStemmer('german')
    res1 = german_stemmer.stem('Überorganisation')
    res2 = german_stemmer.stem('Programmierung')
    res3 = german_stemmer.stem('Einheitsfrontlied')
    res4 = german_stemmer.stem('Studentin')
    res5 = german_stemmer.stem('Erklärung')
    res6 = german_stemmer.stem('Rangordnung')
    res7 = german_stemmer.stem('Ordnung')
    return [res1, res2, res3, res4, res5,res6,res7]
```

8.

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import RegexpStemmer
from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer
def corr_05():
    russian_stemmer = SnowballStemmer('russian')
    w1,w2="кошка","кошачий"
    w3,w4="псевдокошка","окошаченный"
    sw1 = russian_stemmer.stem(w1)
    sw2 = russian_stemmer.stem(w2)
    sw3 = russian_stemmer.stem(w3)
```

```
    sw4 = russian_stemmer.stem(w4)
    print(sw1,sw2,sep='__&__')
    print(sw3,sw4,sep='__&__')
```

9.

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import RegexpStemmer
from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer
def corr_06():
    w1,w2="cooking","cookery"
    w3,w4="cookbooks","feet"
    w5,w6="men","teeth"
    w7,w8="women","geese"
    lemmatizer=WordNetLemmatizer()
    lw1,lw2=lemmatizer.lemmatize(w1),lemmatizer.lemmatize(w2)
    print(lw1,lw2,sep='__&__')
    lw11=lemmatizer.lemmatize(w1,pos='v')
    lw111=lemmatizer.lemmatize(w1,pos='n')
    print(lw11,lw111,sep='__&__')
```

10.

```
def corr_07():
    w="believes"
    stemmer1=PorterStemmer()
    stemmer2=LancasterStemmer()
    stemmer3 = SnowballStemmer('english')
    lemmatizer=WordNetLemmatizer()
    sw1,sw2,sw3=stemmer1.stem(w),stemmer2.stem(w),stemmer3.stem(w)
    lw1=lemmatizer.lemmatize(w,pos='n')
    lw2=lemmatizer.lemmatize(w,pos='v')
    print(lw1,lw2,sep='__&__')
    print(sw1,sw2,sw3,sep='__&__')
```

Индивидуальные задания к разделу 4.

Рассмотреть предложенный пример на встроенном языке Matlab, предназначенный для моделирования.

1. Обучить модель keras распознаванию рукописных букв «R», «U», «V».
2. Обучить модель keras распознаванию рукописных букв «W», «X», «Y».
3. Обучить модель keras распознаванию рукописных букв «G», «Z», «F».
4. Обучить модель keras распознаванию рукописных букв «f», «g», «h».
5. Обучить модель keras распознаванию рукописных букв «i», «j», «k».
6. Обучить модель keras распознаванию рукописных букв «l», «m», «n».
7. Обучить модель keras распознаванию рукописных букв «o», «p», «q».
8. Обучить модель keras распознаванию рукописных букв «r», «s», «t».
9. Обучить модель keras распознаванию рукописных букв «u», «v», «w».
10. Обучить модель keras распознаванию рукописных букв «I», «L», «Q».

7 Оценочные средства для проведения промежуточной аттестации

а) Планируемые результаты обучения и оценочные средства для проведения промежуточной аттестации:

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.		
Знать	– основные принципы анализа информации; основы концепций Data Mining, Text Mining, WEB Scraping; – современные методы мягких вычислений, применяемых при интеллектуальном анализе информации; – современные средства глубокого обучения и методологию их применения.	Список теоретических вопросов: – понятие об информации и ее анализе, информация и данные; – особенности методов интеллектуального анализа информации; – технологии Rules Mining и их применение; – программные средства для анализа текстов, пакет NLTK; – определение статистических характеристик текста; – работа с корпусами текстов, выявление синонимов и антонимов; .
Уметь	– выбирать приемлемые алгоритмы и применять их для решения конкретных задач обработки информации; – самостоятельно конструировать алгоритмы обработки информации в нестандартных ситуациях; – конструировать сложные мультипарадигменные алгоритмы для анализа разнородной и неструктурированной информации.	Список практических заданий: – отследить в Интернете публикации на заданную тему; – выявить среди участников форума ботов (признаками ботов считаются постоянное присутствие, употребление одних и тех же клишированных речевых оборотов).
Владеть	– навыками применения программных средств анализа информации; – навыками настройки сложных систем анализа информации; – навыками разработки программных средств анализа информации	Список комплексных заданий: – отследить в Интернете публикации на несколько взаимосвязанных тем; – отследить группы ботов, присутствующих одновременно на нескольких форумах (имена на каждом форуме у них, скорее всего, будут разные)

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
ОПК-5 Обладает способностью решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.		
Знать	– методологию формальной постановки задачи анализа информации; – методологию анализа и оценки влияния контекста, в котором сформирована информация; – методологию анализа и оценки влияния контекста, в котором функционирует информация.	Список теоретических вопросов: – задачи аннотирования текстов; понятие о WEB Mining; – классификация текстов на основе нейросетей прямого распространения; – классификация текстов на основе нечетких множеств; – классификация текстов с применением нейронечетких сетей.
Уметь	– выбирать концепцию построения модели интеллектуальной системы анализа информации, соответствующую поставленной прикладной задаче; – выбирать алгоритмы верификации функционирования моделей анализа информации.	Список практических заданий: – формализовать задачу нахождения речевых оборотов, набирающих максимальное количество «лайков» в соцсети и реализовать ее с помощью программных средств; – формализовать задачу наличия Product Placement в соцсетях и реализовать ее с помощью программных средств.
Владеть	– навыками применения программного обеспечения интеллектуальных систем для разработки средств анализа информации; – навыками осуществления настройки и верификации программного обеспечения интеллектуальных систем для разработки и функционирования интеллектуальных моделей анализа информации; – навыками осуществления модификации программного обеспечения интеллектуальных систем для разработки и функционирования интеллектуальных моделей	Список комплексных заданий: – определить с помощью нейросетевого анализа различия между тематикой и стилистикой публикаций в соцсетях (или установить их неотличимость); – определить с помощью нейросетевого анализа различия между способами осуществления Product Placement в соцсетях (или установить их неотличимость).

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
	анализа информации.	
ОК-6. Обладает способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия.		
Знать	– основные принципы организации и функционирования микросоциума; – методы предотвращения и разрешения конфликтов; – методологию социального проектирования.	Список теоретических вопросов: – работа с корпусами текстов; – понятие о семантическом анализе текста в рамках широкого контекста.
Уметь	– анализировать состояние коллектива; – находить способы решения конкретных конфликтных ситуаций; – проектировать развитие коллектива в желательном направлении.	Список практических заданий: – выполнить анализ социокультурного контекста нескольких блогов; – выполнить анализ этнического контекста нескольких блогов.
Владеть	– навыками межкультурной коммуникации; – навыками выстраивания системы стабильного развития в коллективе; – навыками оптимального целеполагания для каждого сотрудника и всего коллектива	Список комплексных заданий: – выполнить анализ контента нескольких Интернет-ресурсов в социокультурном, этническом и гендерном контексте; – выполнить анализ контента нескольких Интернет-ресурсов в профессиональном контексте;

б) Порядок проведения промежуточной аттестации, показатели и критерии оценивания:

Промежуточная аттестация по дисциплине «Методы нейрокомпьютерного моделирования» основана на проверке выполнения практических заданий, в ходе которой выявляется степень сформированности умений и владений. Аттестация проводится в форме зачета.

Показатели и критерии оценивания зачета:

– **«зачтено»** – обучающийся демонстрирует сформированность компетенций, умение применять изученный материал в практически важных ситуациях.

– **«не зачтено»** – обучающийся не может показать знания на уровне воспроизведения и объяснения информации, не может показать интеллектуальные навыки решения основных задач.

8 Учебно-методическое и информационное обеспечение дисциплины (модуля)

а) Основная литература:

1. Шитиков В.К. Классификация, регрессия и другие алгоритмы Data Mining с использованием R [Электронный ресурс]./ В.К. Шитиков, С.Э. Мастицкий - Тольятти, Лондон, - 2017, 351 с. Режим доступа: https://ranalytics.github.io/data-mining

б) Дополнительная литература:

1. Радченко И.А. Технологии и инфраструктура Big Data: Учебное пособие. [Электронный ресурс]. / И.А. Радченко, И.Н. Николаев – СПб.: Университет ИТМО, 2018. 55 с. Режим доступа: http://books.ifmo.ru/book/2138/tehnologii_i_infrastructura_Big_Data:_uchebnoe_posobie.htm

в) Методические указания

1. Жидьцов В.В. Практикум по нейросетевым технологиям. [Электронный ресурс]. Режим доступа: http://bek.sibadi.org/fulltext/epd6.pdf

г) Программное обеспечение и Интернет-ресурсы:

Программное обеспечение: лицензионное программное обеспечение: операционная система; офисные программы; математические пакет, статистические пакеты, установленные на каждом персональном компьютере вычислительного центра ФГБОУ ВПО «МГТУ».

Перечень лицензионного программного обеспечения по ссылке:
http://sps.vuz.magtu.ru/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2FShared%20Documents%2F%D0%9F%D0%BE%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B0%20%D0%BA%20%D0%B0%D0%BA%D0%BA%D1%80%D0%B5%D0%B4%D0%B8%D1%82%D0%B0%D1%86%D0%B8%D0%B8%202020%2F%D0%A1%D0%B0%D0%BC%D0%BE%D0%BE%D0%B1%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%202019%D0%B3%2F%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5%20%D0%9F%D0%9E&InitialTabId=Ribbon.Document&VisibilityContext=WSSTabPersistence

Официальные сайты промышленных предприятий и организаций: http://www.mmk.ru, http://www.creditural.ru, http://www.magtu.ru, http://www.gks.ru и т.п.; разработчиков программных продуктов: http://www.statsoft.ru, http://www.microsoft.com, http://www.ptc.com и т.п; сайты лабораторий компьютерной графики http://graphics.cs.msu.ru , http://cgm.graphicon.ru.

9 Материально-техническое обеспечение дисциплины (модуля)

Материально-техническое обеспечение дисциплины включает:

Тип и название аудитории	Оснащение аудитории
Лекционная аудитория	Мультимедийные средства хранения, передачи и представления информации

Тип и название аудитории	Оснащение аудитории
Компьютерный класс	Персональные компьютеры с пакетом Office, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета
Аудитории для самостоятельной работы: компьютерные классы; читальные залы библиотеки	Все классы УИТ и АСУ с персональными компьютерами, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета
Аудиторий для групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации	Ауд. 282 и классы УИТ и АСУ
Помещения для самостоятельной работы обучающихся, оснащенных компьютерной техникой с возможностью подключения к сети «Интернет» и наличием доступа в электронную информационно-образовательную среду организации	Классы УИТ и АСУ
Помещения для хранения и профилактического обслуживания учебного оборудования	Центр информационных технологий – ауд. 379