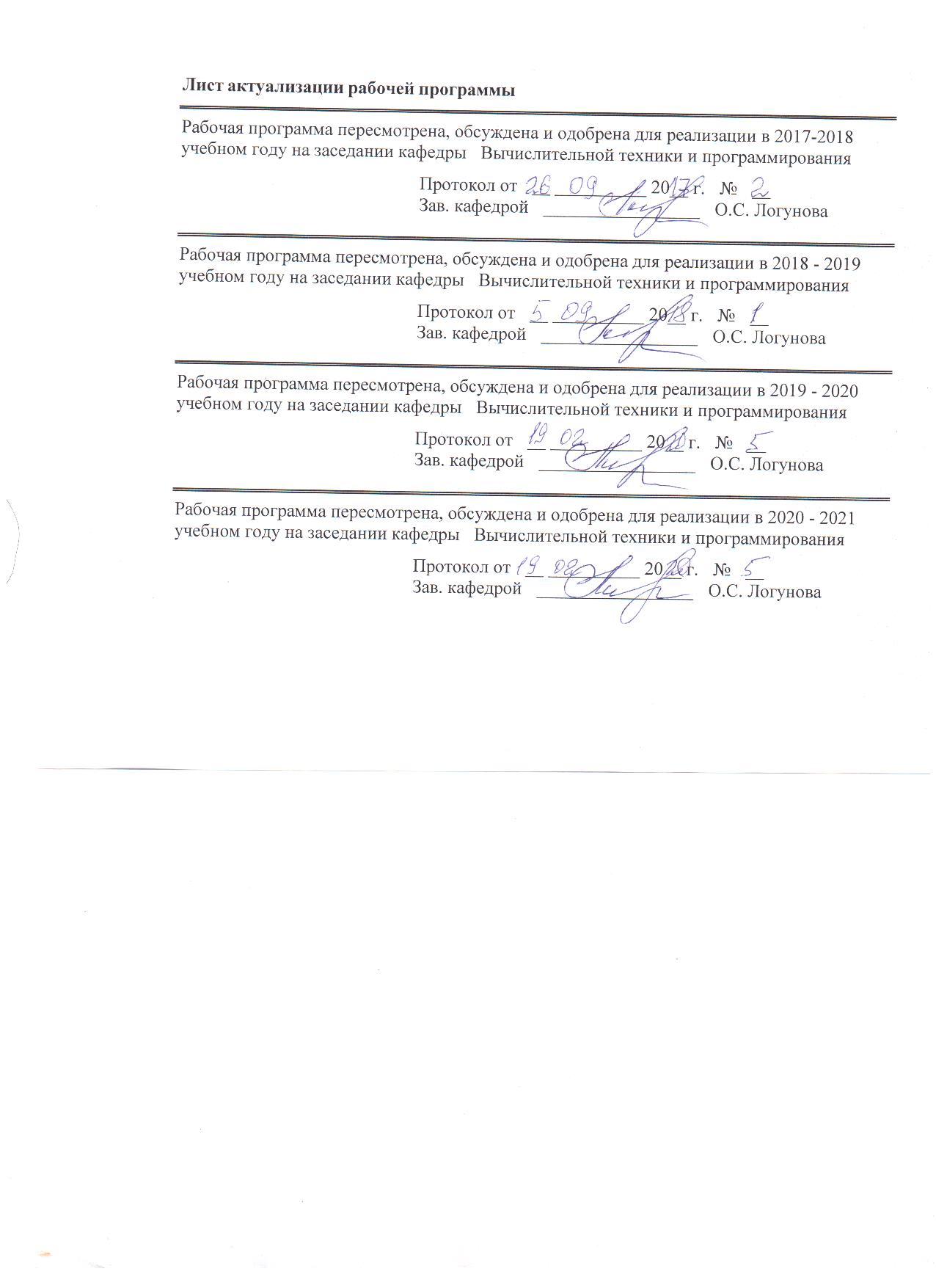


1 Цели освоения дисциплины (модуля)



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; * 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 текстами. | 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* |
| 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 работы. | Беседа – обсуждение.  Анализ программного кода.  Устный опрос. | *ОК-6 ‑ зув*  *ОПК-5 – зув*  *ПК-2 – зув* |
| **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.

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)

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)

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)

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)

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)

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)

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)

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)

def myStr():

"""This is the program myStr"""

s1 = '\74'

print(s1)

s2 = '\x6a'

print(s2)

s3 = '\u043a'

print(s3)

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)

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()

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()

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()

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()

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()

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()

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()

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()

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()

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.

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())

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())

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)

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')

from nltk.book import \*

def Sample\_03():

text1.similar('monstrous')

text2.similar('little')

text3.similar('God')

from nltk.book import \*

def Sample\_04():

text1.common\_contexts(['monstrous', 'very'])

text2.common\_contexts(['little', 'great'])

text3.common\_contexts(['God', 'devil'])

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)

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')

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)

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

def scr():

path=r"http://pythonscraping.com/pages/page1.html"

html=urlopen(path)

print(html.read())

def scr():

path=r"http://pythonscraping.com/pages/page1.html"

html=urlopen(path)

bsObj=BeautifulSoup(html.read())

print(bsObj.h1)

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())

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

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)

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())

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))

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())

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'])

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

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('-------------------------------------------')

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('-------------------------------------------')

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('-------------------------------------------')

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('-------------------------------------------')

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('-------------------------------------------')

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('-------------------------------------------')

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('-------------------------------------------')

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('-------------------------------------------')

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('-------------------------------------------')

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.

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='\_\_&\_\_')

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='\_\_&\_\_')

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)

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)

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)

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

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())

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)

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)

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

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)

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)

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)

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)

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)

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)

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)

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)

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)

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.

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))

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)

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')

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)

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)

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)

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)

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)

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)

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.

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 current\_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]

def stFuzz\_11():

"""This is a simple program which uses fuzzy technologies"""

key="Котов Василий"

choices=['Котов Васiль','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)

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)

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)

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)

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='\_\_&\_\_')

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]

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='\_\_&\_\_')

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='\_\_&\_\_')

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) |
| **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 анализа информации. | 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). |
| **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 |
| 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 |