Московский государственный технический университет им. Н.Э. Баумана

Кафедра ИУ-5 «Системы обработки информации и управления»

**Отчет по лабораторной работе №7**

по курсу «Проектирование интеллектуальных систем»

«Использование нейронных сетей для анализа текста»

|  |  |
| --- | --- |
| Выполнил: |  |
| студент группы ИУ5-23М |  |
| Березин Иван | "\_\_"\_\_\_\_\_\_\_\_\_2019 г |
| Преподаватель: |  |
| Терехов В.И., |  |
| к.т.н., доцент | "\_\_"\_\_\_\_\_\_\_\_\_2019 г |

Москва, 2019

**Цель работы:**

Научиться предобрабатывать текстовые данные для использования в нейросетях и научиться использовать нейронные сети для классификации текстов.

**Задание:**

Необходимо увеличить количество скрытых слоев в описанной нейронной сети до 3-ех, а количество нейронов увеличить таким образом, чтобы точность работы нейронной сети составляла не менее 75%. Темы текстов изменить в соответствии с вариантом.

|  |  |
| --- | --- |
| Вариант | Темы текстов |
| 3 | talk.politics.guns, alt.atheism, sci.med, rec.autos |

**Выполнение:**

|  |
| --- |
| import numpy as np  import tensorflow as tf  from collections import Counter  from sklearn.datasets import fetch\_20newsgroups |

**Выбор категорий текстов, добавление в словари.**

|  |
| --- |
| categories = ["comp.graphics", "rec.sport.hockey", "sci.crypt", "talk.politics.guns"]  newsgroups\_train = fetch\_20newsgroups(subset='train', categories=categories)  newsgroups\_test = fetch\_20newsgroups(subset='test', categories=categories)  vocab = Counter()  for text in newsgroups\_train.data:  for word in text.split(' '):  vocab[word.lower()] += 1    for text in newsgroups\_test.data:  for word in text.split(' '):  vocab[word.lower()] += 1    total\_words = len(vocab) |

**Коллекция word2index**

|  |
| --- |
| def get\_word\_2\_index(vocab):  word2index = {}  for i, word in enumerate(vocab):  word2index[word.lower()] = i  return word2index  word2index = get\_word\_2\_index(vocab) |

**Функция получения батча**

|  |
| --- |
| def get\_batch(df, i, batch\_size):  batches = []  results = []  texts = df.data[i \* batch\_size:i \* batch\_size + batch\_size]  categories = df.target[i \* batch\_size:i \* batch\_size + batch\_size]  for text in texts:  layer = np.zeros(total\_words, dtype=float)  for word in text.split(' '):  layer[word2index[word.lower()]] += 1  batches.append(layer)  for category in categories:  y = np.zeros((4), dtype=float)  if category == 0:  y[0] = 1.  elif category == 1:  y[1] = 1.  elif category == 2:  y[2] = 1.  else:  y[3] = 1.  results.append(y)    return np.array(batches), np.array(results) |

**Настройка параметров нейросети.**

|  |
| --- |
| # Параметры обучения  learning\_rate = 0.01  training\_epochs = 10  batch\_size = 150  display\_step = 1  # Network Parameters  n\_hidden\_1 = 130 # скрытый слой  n\_hidden\_2 = 150 # скрытый слой  n\_hidden\_3 = 100 # скрытый слой  n\_input = total\_words # количество уникальных слов в наших текстах  n\_classes = 4 # 4 класса  input\_tensor = tf.placeholder(tf.float32,[None, n\_input],name="input")  output\_tensor = tf.placeholder(tf.float32,[None, n\_classes],name="output") |

**Описание структуры нейросети**

|  |
| --- |
| def multilayer\_perceptron(input\_tensor, weights, biases):  # скрытый слой  layer\_1\_multiplication = tf.matmul(input\_tensor, weights['h1'])  layer\_1\_addition = tf.add(layer\_1\_multiplication, biases['b1'])  layer\_1 = tf.nn.relu(layer\_1\_addition)  # скрытый слой  layer\_2\_multiplication = tf.matmul(layer\_1, weights['h2'])  layer\_2\_addition = tf.add(layer\_2\_multiplication, biases['b2'])  layer\_2 = tf.nn.relu(layer\_2\_addition)    # скрытый слой  layer\_3\_multiplication = tf.matmul(layer\_2, weights['h3'])  layer\_3\_addition = tf.add(layer\_3\_multiplication, biases['b3'])  layer\_3 = tf.nn.relu(layer\_3\_addition)  # выходной слой  out\_layer\_multiplication = tf.matmul(layer\_3, weights['out'])  out\_layer\_addition = out\_layer\_multiplication + biases['out']    return out\_layer\_addition |

|  |
| --- |
| # инициализация параметров сети  weights = {  'h1': tf.Variable(tf.random\_normal([n\_input, n\_hidden\_1])),  'h2': tf.Variable(tf.random\_normal([n\_hidden\_1, n\_hidden\_2])),  'h3': tf.Variable(tf.random\_normal([n\_hidden\_2, n\_hidden\_3])),  'out': tf.Variable(tf.random\_normal([n\_hidden\_3, n\_classes]))  }  biases = {  'b1': tf.Variable(tf.random\_normal([n\_hidden\_1])),  'b2': tf.Variable(tf.random\_normal([n\_hidden\_2])),  'b3': tf.Variable(tf.random\_normal([n\_hidden\_3])),  'out': tf.Variable(tf.random\_normal([n\_classes]))  }  # создание модели  prediction = multilayer\_perceptron(input\_tensor, weights, biases)  # Фукнция потерь  loss = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits(logits=prediction, labels=output\_tensor))  optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate).minimize(loss)  init = tf.global\_variables\_initializer()  # Запуск  with tf.Session() as sess:  sess.run(init)  # Цикл обучения  for epoch in range(training\_epochs):  avg\_cost = 0.  total\_batch = int(len(newsgroups\_train.data)/batch\_size)  # Проход по всем батчам  for i in range(total\_batch):  batch\_x,batch\_y = get\_batch(newsgroups\_train,i,batch\_size)  c,\_ = sess.run([loss,optimizer], feed\_dict={input\_tensor: batch\_x,output\_tensor:batch\_y})  # Вычисляем среднее фукнции потерь  avg\_cost += c / total\_batch  print("Эпоха:", '%04d' % (epoch+1), "loss=", "{:.16f}".format(avg\_cost))  print("Обучение завершено!")  # Тестирование  correct\_prediction = tf.equal(tf.argmax(prediction, 1), tf.argmax(output\_tensor, 1))  # Расчет точности  accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, "float"))  total\_test\_data = len(newsgroups\_test.target)  batch\_x\_test,batch\_y\_test = get\_batch(newsgroups\_test,0,total\_test\_data)  print("Точность:", accuracy.eval({input\_tensor: batch\_x\_test, output\_tensor: batch\_y\_test}))  C:\Users\uivan_000\Desktop\Снимок.JPG |

**Список литературы**

1. Документация по tensorflow. https://www.tensorflow.org/.

2. Глубокое обучение для NLP. https://nlp.stanford.edu/courses/NAACL2013/