**Московский государственный технический университет им. Н.Э. Баумана**

Кафедра «Системы обработки информации и управления»

**Лабораторная работа №7**

**по курсу «Проектирование интеллектуальных систем»**

**Выполнил:**

**Хотин П.Ю.**

**ИУ5-24М**

Москва, 2020 год

**Выполнение задачи**

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

categories = ["comp.windows.x", "rec.motorcycles", "sci.crypt", "sci.space"]  
newsgroups\_train = fetch\_20newsgroups(subset='train', categories=categories)  
newsgroups\_test = fetch\_20newsgroups(subset='test', categories=categories)

len(newsgroups\_train.target)

2379

print('total texts in train:', len(newsgroups\_train.data))  
print('total texts in test:', len(newsgroups\_test.data))  
print('categories:', newsgroups\_train.target, newsgroups\_train.target\_names)

total texts in train: 2379  
total texts in test: 1583  
categories: [2 2 3 ... 3 1 0] ['comp.windows.x', 'rec.motorcycles', 'sci.crypt', 'sci.space']

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)

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)  
  
  
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)

# Параметры обучения  
learning\_rate = 0.01  
training\_epochs = 16  
batch\_size = 140  
display\_step = 1  
  
# Network Parameters  
n\_hidden\_1 = 420 # скрытый слой  
n\_hidden\_2 = 210 # скрытый слой  
n\_hidden\_3 = 90  
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")

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}))

Эпоха: 0001 loss= 18617.0064086914062500  
Эпоха: 0002 loss= 4563.1136188507080078  
Эпоха: 0003 loss= 3242.6922111511230469  
Эпоха: 0004 loss= 4597.3344640731811523  
Эпоха: 0005 loss= 3107.6046485900878906  
Эпоха: 0006 loss= 577.2595361173152924  
Эпоха: 0007 loss= 238.7371116876602173  
Эпоха: 0008 loss= 9.4245774447917938  
Эпоха: 0009 loss= 5.2540618411730975  
Эпоха: 0010 loss= 0.0000000011175857  
Эпоха: 0011 loss= 0.0000000000000000  
Эпоха: 0012 loss= 0.0000000000000000  
Эпоха: 0013 loss= 0.0000000000000000  
Эпоха: 0014 loss= 0.0000000000000000  
Эпоха: 0015 loss= 0.0000000000000000  
Эпоха: 0016 loss= 0.0000000000000000  
Обучение завершено!  
Точность: 0.75868607

**Ответы на вопросы**

1. **Какие вы знаете задачи обработки текстов, в чем они заключаются?**

Классификация(выявление отношения к группам), кластеризация(выделение из групп текстов одинаковой тематики), построение ассоциативных правил(закономерности между словами), машинный перевод.

1. **Зачем нужна предобработка текста для машинного обучения?**

Предобработка текста переводит текст на естественном языке в формат удобный для дальнейшей работы. Ключевую роль играют такие факторы как, порядок слов, наличие словоформы.

1. **Какие виды предобработки текста вы знаете?**

Перевод всех букв в тексте в нижний или верхний регистры;

Удаление пробельных символов (whitespaces);

Удаление стоп слов;

Стемминг.

1. **Что такое стемминг?**

Количество корректных словоформ, значения которых схожи, но написания отличаются суффиксами, приставками, окончаниями и прочим, очень велико, что усложняет создание словарей и дальнейшую обработку. Стемминг позволяет привести слово к его основной форме.

1. **Что такое 20 Newsgroups?**

Набор из 18000 статей, разделенных между собой на 20 тематик, каждая из которых относится либо к тренировочной, либо к тестовой выборкам.

1. **Чему должно равняться число входных и выходных нейронов в задаче классификации текстов?**

Число входных нейронов должно равняться числу уникальных слов в тексте, а число выходных нейронов должно соответствовать количество классов.

**Список литературы**

##### [1] Google. Tensorflow. 2018. Apr. url - https://www.tensorflow.org/api\_docs/python/tf/train/Saver.

##### [2] Google. TensorBoard. 2018. Apr. url - https://www.tensorflow.org/programmers\_guide/summaries\_and\_tensorboard.