Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

|  |
| --- |
|  |

**ЛабоРАТОРНАЯ РАБОТА №7**

«Рекуррентные нейронные сети для анализа текста»

|  |  |  |
| --- | --- | --- |
| Студент |  | К. Д. Зюсько |
| Преподаватель |  | М. В. Стержанов |

Минск 2020

ХОД РАБОТЫ

**Задание.**

Набор данных для предсказания оценок для отзывов, собранных с сайта imdb.com, который состоит из 50,000 отзывов в виде текстовых файлов. Отзывы разделены на положительные (25,000) и отрицательные (25,000). Данные предварительно токенизированы по принципу “мешка слов”, индексы слов можно взять из словаря (imdb.vocab). Обучающая выборка включает в себя 12,500 положительных и 12,500 отрицательных отзывов, контрольная выборка также содержит 12,500 положительных и 12,500 отрицательных отзывов, а также. Данные можно скачать на сайте Kaggle -> https://www.kaggle.com/iarunava/imdb-movie-reviews-dataset

1. Загрузите данные. Преобразуйте текстовые файлы во внутренние структуры данных, которые используют индексы вместо слов.
2. Реализуйте и обучите двунаправленную рекуррентную сеть (LSTM или GRU). Какого качества классификации удалось достичь?
3. Используйте индексы слов и их различное внутреннее представление (word2vec, glove). Как влияет данное преобразование на качество классификации?
4. Поэкспериментируйте со структурой сети (добавьте больше рекуррентных, полносвязных или сверточных слоев). Как это повлияло на качество классификации?
5. Используйте предобученную рекуррентную нейронную сеть (например, DeepMoji или что-то подобное).

**Результат выполнения:**

Поскольку в данной лабораторной предлагается сравнить результаты нескольких различных конфигураций нейронных сетей, я установил количество эпох для обучение равным 10, чтобы условия, в которых происходит сравнение, были одинаковыми.

1. Код реализации:

!mkdir ~/.kaggle

!cp /content/.kaggle/kaggle.json ~/.kaggle/kaggle.json

!kaggle datasets download lakshmi25npathi/imdb-dataset-of-50k-movie-reviews -p /content/kaggle/imdb

!unzip kaggle/imdb/imdb-dataset-of-50k-movie-reviews.zip -d data

import pandas as pd

df = pd.read\_csv('data/IMDB Dataset.csv')

print(df.shape)

df.head()

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM

from tensorflow.keras.layers import SimpleRNN

from tensorflow.keras.layers import Embedding

from tensorflow.keras.layers import Bidirectional

from tensorflow.keras.layers import BatchNormalization

from tensorflow.keras.layers import Dropout

from tensorflow.keras.utils import to\_categorical

from sklearn.utils import shuffle

import matplotlib.pyplot as plt

dictionary\_length = 10000

input\_length = 100

tokenizer = Tokenizer(num\_words=dictionary\_length)

tokenizer.fit\_on\_texts(df.review.values)

post\_seq = tokenizer.texts\_to\_sequences(df.review.values)

word\_index = tokenizer.word\_index

post\_seq\_padded = pad\_sequences(post\_seq, maxlen=input\_length)

import numpy as np

x\_original = post\_seq\_padded

x\_original = np.array(x\_original)

y\_original = df['sentiment'].replace({ 'positive': 1, 'negative': 0 }).values

y\_original = np.array(y\_original)

x, y = shuffle(x\_original, y\_original, random\_state=23)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.5, random\_state=42)

x\_train, x\_val, y\_train, y\_val = train\_test\_split(x\_train, y\_train, test\_size=0.2, random\_state=42)

2. Код модели:

model = Sequential()

model.add(Embedding(dictionary\_length, 8, input\_length=input\_length))

model.add(Bidirectional(LSTM(16, return\_sequences=False))) # dropout=0.2, recurrent\_dropout=0.2

model.add(Dense(1, activation='sigmoid'))

model.summary()

Результаты обучения приведены в таблице ниже.

Таблица 1 – результаты обучения

|  |  |  |
| --- | --- | --- |
|  | Loss | Accuracy |
| Тренировочный датасет | 0.0912 | 97.32% |
| Валидационный датасет | 0.5607 | 83.20% |
| Тестовый датасет | 0.5428 | 83.40% |

Как можно видеть выше, ошибка на валидационном датасете в несколько раз больше, чем на тренировочном. Поэтому здесь очевидна проблема high variance.

3. Код реализации:

!wget <http://nlp.stanford.edu/data/glove.6B.zip>

!mkdir glove

!unzip glove.6B.zip -d glove

glove\_dir = 'glove'

embeddings\_index = {}

f = open(os.path.join(glove\_dir, 'glove.6B.100d.txt'))

for line in f:

values = line.split()

word = values[0]

coefs = np.asarray(values[1:], dtype='float32')

embeddings\_index[word] = coefs

f.close()

embedding\_dim = 100

embedding\_matrix = np.zeros((dictionary\_length, embedding\_dim))

for word, i in word\_index.items():

embedding\_vector = embeddings\_index.get(word)

if i < dictionary\_length:

if embedding\_vector is not None:

# Words not found in embedding index will be all-zeros.

embedding\_matrix[i] = embedding\_vector

model = Sequential()

model.add(Embedding(dictionary\_length, embedding\_dim, input\_length=input\_length))

model.add(Bidirectional(LSTM(16))) # dropout=0.2, recurrent\_dropout=0.2

model.add(Dense(1, activation='sigmoid'))

model.summary()

model.layers[0].set\_weights([embedding\_matrix])

model.layers[0].trainable = False

# try using different optimizers and different optimizer configs

model.compile(loss='binary\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

model.fit(x\_train, y\_train, batch\_size=32, epochs=10, verbose=1, validation\_data=(x\_val, y\_val))

score, acc = model.evaluate(x\_test, y\_test)

print('Test accuracy:', acc)

Таблица 2 – результаты обучения

|  |  |  |
| --- | --- | --- |
|  | Loss | Accuracy |
| Тренировочный датасет | 0.2831 | 87.73% |
| Валидационный датасет | 0.3467 | 84.38% |
| Тестовый датасет | 0.3442 | 84.90% |

Из таблицы выше видно, что ошибки отличаются друг от друга, но не с таким большим разбегом как раньше, что свидетельствует о том, что проблема high variance решена.

4. Здесь я добавил два рекурентных слоя, и несколько полносвязных. В качестве embedding слоя используется GloVe. Код реализации:

model = Sequential()

model.add(Embedding(dictionary\_length, embedding\_dim, input\_length=input\_length))

model.add(Bidirectional(LSTM(32, return\_sequences=True))) # dropout=0.2, recurrent\_dropout=0.2

model.add(Bidirectional(LSTM(32, return\_sequences=True)))

model.add(Bidirectional(LSTM(32)))

model.add(Dense(128, activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.summary()

model.layers[0].set\_weights([embedding\_matrix])

model.layers[0].trainable = False

# try using different optimizers and different optimizer configs

model.compile(loss='binary\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

model.fit(x\_train, y\_train, batch\_size=32, epochs=10, verbose=1, validation\_data=(x\_val, y\_val))

score, acc = model.evaluate(x\_test, y\_test)

print('Test accuracy:', acc)

Таблица 3 – результаты обучения

|  |  |  |
| --- | --- | --- |
|  | Loss | Accuracy |
| Тренировочный датасет | 0.2073 | 91.81% |
| Валидационный датасет | 0.4219 | 83.54% |
| Тестовый датасет | 0.4177 | 83.56% |

В данном случае увеличение слоёв не дало значительного улучшеиня качества модели.

4. Здесь я добавил два рекурентных слоя, и несколько полносвязных. В качестве embedding слоя используется GloVe. Код реализации:

!git clone <https://github.com/bfelbo/DeepMoji.git>

%cd DeepMoji

import sys

from os.path import abspath, dirname

sys.path.insert(0, '/content/DeepMoji')

sys.path.insert(0, '/content/DeepMoji/deepmoji')

sys.path.insert(0, '/content/DeepMoji/examples')

print(sys.path)

%cd scripts

from \_\_future\_\_ import print\_function

import os

from subprocess import call

curr\_folder = os.path.basename(os.path.normpath(os.getcwd()))

weights\_filename = 'deepmoji\_weights.hdf5'

weights\_folder = 'model'

weights\_path = '{}/{}'.format(weights\_folder, weights\_filename)

if curr\_folder == 'scripts':

weights\_path = '../' + weights\_path

weights\_download\_link = 'https://www.dropbox.com/s/xqarafsl6a8f9ny/deepmoji\_weights.hdf5?dl=0#'

MB\_FACTOR = float(1 << 20)

def prompt():

while True:

valid = {

'y': True,

'ye': True,

'yes': True,

'n': False,

'no': False,

}

if 'TRAVIS' in os.environ:

choice = 'yes'

else:

choice = input().lower()

if choice in valid:

return valid[choice]

else:

print('Please respond with \'y\' or \'n\' (or \'yes\' or \'no\')')

download = True

if os.path.exists(weights\_path):

print('Weight file already exists at {}. Would you like to redownload it anyway? [y/n]'.format(weights\_path))

download = prompt()

already\_exists = True

else:

already\_exists = False

if download:

print('About to download the pretrained weights file from {}'.format(weights\_download\_link))

if not already\_exists:

print('The size of the file is roughly 85MB. Continue? [y/n]')

else:

os.unlink(weights\_path)

if already\_exists or prompt():

print('Downloading...')

# urllib.urlretrieve(weights\_download\_link, weights\_path)

# with open(weights\_path,'wb') as f:

# f.write(requests.get(weights\_download\_link).content)

# downloading using wget due to issues with urlretrieve and requests

sys\_call = 'wget {} -O {}'.format(weights\_download\_link, os.path.abspath(weights\_path))

print("Running system call: {}".format(sys\_call))

call(sys\_call, shell=True)

if os.path.getsize(weights\_path) / MB\_FACTOR < 80:

raise ValueError("Download finished, but the resulting file is too small! " +

"It\'s only {} bytes.".format(os.path.getsize(weights\_path)))

print('Downloaded weights to {}'.format(weights\_path))

else:

print('Exiting.')

!pip3 uninstall -y tensorflow

!pip3 install tensorflow==1.13.1

from \_\_future\_\_ import print\_function

import numpy as np

from keras.preprocessing import sequence

from keras.datasets import imdb

from deepmoji.model\_def import deepmoji\_architecture

# Seed for reproducibility

np.random.seed(1337)

batch\_size = 256

print('Build model...')

model = deepmoji\_architecture(nb\_classes=2, nb\_tokens=dictionary\_length, maxlen=input\_length)

model.summary()

model.compile(loss='binary\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

print('Train...')

model.fit(x\_train, y\_train, batch\_size=batch\_size, epochs=10,

validation\_data=(x\_val, y\_val))

score, acc = model.evaluate(x\_test, y\_test, batch\_size=batch\_size)

print('Test score:', score)

print('Test accuracy:', acc)

Таблица 4 – результаты обучения

|  |  |  |
| --- | --- | --- |
|  | Loss | Accuracy |
| Тренировочный датасет | 0.1987 | 85.66% |
| Валидационный датасет | 0.2081 | 85.12% |
| Тестовый датасет | 0.2057 | 85.23% |

Данная модель показала наилучший результат.