Методические указания по выполнению лабораторной работы №4
«Распознавание последовательностей»

Данная лабораторная работа посвящена теме обработки данных, представляемых в виде последовательностей. Особенность таких данных заключается в том, что очередное значение последовательности некоторым образом зависит от одного или нескольких соседних значений и из этих скрытых зависимостей в можно выявлять дополнительные свойства данных. Примером данных, представляемых в виде последовательностей, могут быть предложения и тексты на естественном языке. Представьте, что перед вами стоит задача построения модели, которая по некоторому фрагменту текста должна предсказать следующее слово:

Питер каждое лето проводил во Франции у бабушки, поэтому он хорошо говорит на …

Наша биологическая нейронная сеть за несколько первых слов предложения уловила контекст и легко подсказывает нам, что следует закончить предложение словом *«французском»*. Но как перенести это улавливание контекста в модель? Для этого могут использоваться рекуррентные искусственные нейронные сети. Такие сети отличаются тем, что данные при обработке передаются не только в прямом направлении от входа к выходу, но также могут с сниматься выхода отдельных нейронов или всей сети, преобразовываться и также подаваться на вход. За счет этого становится возможным то самое улавливание контекста и выявление скрытых зависимостей в данных.

Другими примерами последовательных данных и задач могут быть временные ряды (предсказание последовательностей, прогнозы, обнаружение аномалий) или кадры в видео (восстановление пропущенных кадров по соседним, повышение качества видео). Задачи обработки текста тоже могут быть различными: можно предсказывать очередное слово (сейчас это используют в клавиатурах мобильных устройств или при выводе подсказок в поисковой строке) или делать машинный перевод. Одной из задач обработки текста является задача определения тональности (sentiment analysis). Фактически, это задача классификации, где объектом является текст, который надо отнести к одному из классов. Может ставиться задача бинарной классификации (позитивный или негативный твит) или многоклассовой (отзыв с отметкой «отлично», «хорошо», «средне», «плохо», «ужасно»).

В данной работе мы попробуем решить задачу определения тональности отзыва на фильм. В качестве выборки возьмем набор данных IMDb (Internet Movie Database), состоящий из 50000 англоязычных отзывов на фильмы, размеченных на два класса: позитивный и негативный.

Начало работы, подключение ускорителя GPU

Обучение рекуррентных нейронных сетей – вычислительно сложная задача, поэтому для ускорения процесса можно перенести вычисления на аппаратный ускоритель на основе графического процессора (GPU).

Для этого в верхнем меню блокнота требуется выбрать «Изменить (Edit)» – «Настройки блокнота (Notebook settings)» – выбрать «GPU» из выпадающего списка «Аппаратный ускоритель (Hardware accelerator)» – «Сохранить (Save)». После сохранения настроек среда выполнения автоматически перезапустится. Для проверки подключения графического ускорителя можно воспользоваться следующим кодом:

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

Корректным выводом будет «`Found GPU at: /device:GPU:0`»

Загрузка набора данных

В Keras есть встроенные функции для загрузки и работы с набором данных IMDb. Загрузка набора данных может быть выполнена следующим образом:

```
# загрузка датасета
from keras.datasets import imdb

vocabulary_size = 5000
index_from = 3

(X_train, y_train), (X_test, y_test) = imdb.load_data(path="imdb.npz",
                                                      num_words=vocabulary_size,
                                                      skip_top=0,
                                                      maxlen=None,
                                                      seed=123,
                                                      start_char=1,
                                                      oov_char=2,
                                                      index_from=index_from
                                                      )
```

Разберем функцию imdb.load_data. Она загружает набор данных вместе с метками классов для каждого отзыва и разбивает его на обучающие и тестовые массивы. Особенность функции в том, что загружаемые отзывы уже прошли первый этап предобработки. Каждый отзыв представляется не в виде строки текста, а в виде списка индексов слов. Для получения этих индексов предварительно весь корпус текстов был разбит на отдельные слова, слова приведены к нижнему регистру, удалены знаки

препинания. Затем слова были упорядочены по частоте встречаемости в корпусе текстов и порядковые номера были приняты в качестве индексов, то есть более частые слова типа «a», «the» имеют меньший индекс, а более редкие слова (имена собственные, слова с опечатками) имеют больший индекс.

Всего в корпусе текстов IMDb встречается более 88000 уникальных слов. Часто для анализа не нужны все слова, так как подавляющее большинство из них встречается крайне редко или слишком часто (слова с малыми и большими индексами не информативны). Для управления загрузкой есть параметр num_words – он определяет размер используемого словаря, по умолчанию имеет значение None, что означает использовать весь словарь (более 88000 слов) и параметр skip_top – позволяет пропустить некоторое количество самых частых слов.

Также при переводе слов в индексы используются некоторые спецсимволы. Например, каждый отзыв начинается с символа начала последовательности, по умолчанию он имеет индекс «1», за этот индекс отвечает параметр start_char. Параметр oov_char (out of vocabulary) определяет то, какой индекс получит слово, не попавшее в словарь из-за ограничения num_words и skip_top.

Параметр maxlen определяет максимальную длину отзыва. Более длинные отзывы будут обрезаны до максимальной длины. Рекомендуется оставить значение None, так как приведение отзывов к единой длине будет осуществляться позже.

Параметр index_from определяет, какой индекс будет иметь первое самое частое слово после всех спецсимволов. Значение по умолчанию «3».

Параметр seed задает инициализирующее значение для случайного разбиения отзывов на обучающие и тестовые для получения воспроизводимых разбиений.

Помимо загрузки набора данных Keras позволяет загрузить словарь пар «слово:индекс» с помощью функции get_word_index. Этот словарь может использоваться для перевода отзыва в виде списка индексов обратно в текст. Однако загружаемый словарь не учитывает спецсимволы и самое частое слово имеет индекс «1». Для корректного перевода индексов в слова нужно сдвинуть все индексы на значение параметра index_from, и добавить в словарь спецсимволы и их индексы. После того, как словарь будет уточнен, необходимо его обратить и получить словарь пар «индекс:слово». Для совершения указанных операций может быть выполнен следующий код:

```
# создание словаря для перевода индексов в слова
# заргузка словаря "слово:индекс"
word_to_id = imdb.get_word_index()

# уточнение словаря
word_to_id = {key:(value + index_from) for key,value in word_to_id.items()}
word_to_id["<PAD>"] = 0
word_to_id["<START>"] = 1
word_to_id["<UNK>"] = 2
word_to_id["<UNUSED>"] = 3

# создание обратного словаря "индекс:слово"
id_to_word = {value:key for key,value in word_to_id.items()}
```

Преобразование отзыва в текст, определение максимальной длины отзыва

Для преобразования отзыва в виде списка индексов в текст может быть выполнен следующий код:

```
review_as_text = ' '.join(id_to_word[id] for id in X_train[some_number])
```

Для определения максимальной длины отзыва в обучающем множестве может быть использовано следующее выражение:

```
len(max(X_train, key=len))
```

Предобработка данных

Предобработка данных будет заключаться в приведении всех отзывов к единой длине. Для этого используется функция Keras pad_sequences из модуля preprocessing.sequence. Она принимает на вход массив списков индексов различной длины (в таком виде изначально представлены X_train и X_test) и выдает на выходе двумерный массив размера (количество примеров, максимальная длина примера). Также входным параметром является собственно длина maxlen и параметр value. В результате обработки массива этой функцией длинные отзывы будут обрезаны до длины maxlen, а короткие дополнены до длины maxlen значениями value. Также дополнительными параметрами являются padding и truncating, которые принимают значения ‘pre’ или ‘post’. Данные параметры определяют, с начала или с конца последовательности будут дополнены и обрезаны. Рекомендуемое значение параметра padding = ‘pre’.

Для предобработки массива X_train может быть использован следующий код:

```
# предобработка данных
from tensorflow.keras.utils import pad_sequences

max_words = 500
X_train = pad_sequences(X_train, maxlen=max_words, value=0, padding='pre', trunca
ting='post')
```

Реализация рекуррентной нейронной сети и оценка качества классификации

После предобработки каждый объект обучающей и тестовой выборки представляет собой вектор фиксированной длины, состоящий из индексов слов. При этом индексы лежат в диапазоне от единиц до тысяч или десятков тысяч (до объема словаря). Из данных в таком виде сложно выявить какие-то зависимости. Для решения этой проблемы при обработке текстов, обычно первым слоем в нейронную сеть добавляют слой Embedding. Данный слой реализует преобразование по принципу word2vec, то есть каждое слово, представленное в виде целочисленного индекса, переводится в вектор фиксированной длины. При обучении весовые коэффициенты преобразования настраиваются так, чтобы в некотором смысле близкие слова переводились в близкие вектора, а далекие – в далекие.

Параметрами слоя Embedding, среди прочих, являются input_dim – размер словаря, output_dim – размер выходного векторного представления и параметр input_length – длина входного вектора (установленная на предыдущем шаге единая длина отзыва). Рекомендуется выбрать output_dim порядка 16…32.

Далее рекомендуется добавить в сеть слой рекуррентных LSTM-ячеек. Для слоя LSTM достаточно указать параметр units порядка 50…100, а остальные параметры оставить по умолчанию.

Далее рекомендуется добавить в сеть слой Dropout. Для слоя Dropout достаточно указать параметр rate порядка 0.2…0.5, а остальные параметры оставить по умолчанию.

Последним слоем в сеть добавляется слой Dense. Так как решается задача бинарной классификации и выходные значения y_train и y_test представлены просто как 0 или 1, то в выходной слой достаточно добавить 1 нейрон с функцией активации ‘sigmoid’.

Далее модель необходимо скомпилировать, указав функцию потерь loss (рекомендуется выбрать ‘binary_crossentropy’), оптимизатор optimizer (рекомендуется ‘adam’) и список метрик, по которым будет оцениваться качество обучения.

При запуске процесса обучения требуется выделить часть обучающих данных под валидацию. За объем валидационного множества отвечает параметр validation_split метода fit. Также при обучении рекомендуется выбрать размер батча batch_size порядка 32…64, и количество эпох epochs 3…5.

После завершения обучения требуется оценить качество классификации на тестовой выборке с помощью метода evaluate.

Для вывода отчета о качестве классификации тестовой выборки можно воспользоваться функцией classification_report из пакета sklearn. Этой функции необходимо передать массив истинных меток классов и массив предсказанных меток классов. Так как модель на выходе дает одно число от 0 до 1 (один выходной нейрон с функцией активации sigmoid), то предсказанную метку можно определить по превышению порога, например, 0.5.

```
y_score = model.predict(X_test)
y_pred = [1 if y_score[i,0]>=0.5 else 0 for i in range(len(y_score))]

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, labels = [0, 1], target_names=['Negat
ive', 'Positive']))
```

Для построения ROC-кривой не требуется переводить ответы нейронной сети в бинарные предсказания по какому-либо порогу. Используются «сырые» выходы модели из интервала от 0 до 1. При использовании библиотек skleran и matplotlib, построение ROC-кривой и вычисление площади под ней может быть выполнено с помощью следующего кода (используются функции roc_curve и auc):

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr, tpr, thresholds = roc_curve(y_test, y_score)
plt.plot(fpr, tpr)
plt.grid()
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
plt.show()
print('Area under ROC is', auc(fpr, tpr))
```

Обратите внимание, в функцию roc_curve необходимо передавать не бинарный ответ модели `y_pred`, который мы сами определили по некоторому порогу, а сырой выход модели `y_score`.

Подробнее про ROC и AUC ROC: Википедия, Google ML course.

Вычислить AUC ROC также можно и без построения графика ROC, вызвав функцию roc_auc_score из sklearn:

```
from sklearn.metrics import roc_auc_score
print('AUC ROC:', roc_auc_score(y_test, y_score))
```