Таким образом, для повышения квантового выхода антистоксовой люминесценции рассматриваемых стекол наряду с удовлетворением отмеченным выше требованиям немаловажным представляется и поиск систем, обеспечивающих наименьшие расстояния сближения между ионами Yb^{3+} и Er^{3+}. Естественно также, что при прочих равных условиях предпочтение будут иметь стекла, обладающие более высокими значениями спектрального коэффициента Эйнштейна для абсорбционного перехода $^2F_{7/2} \rightarrow {}^2F_{5/2}$ ионов Yb^{3+}.

С точки зрения реализации малых $R(Yb-Er)$ интерес представляют стекла системы TeO_2–WO_3–Ln_2O_3, обеспечивающие $R(Ln-Ln) \sim 3{,}8$–$4{,}0$Å [4]. Немаловажным представляется и то обстоятельство, что отсутствие в этих стеклах щелочных металлов обеспечивает повышенные значения эффективной силы химических связей структурного каркаса, поскольку увеличение этого параметра сопровождается уменьшением $R(Ln–Ln)$, возрастанием $R(Ln-O)$, уширением спектральных полос ионов лантаноидов и их смещением в коротковолновую сторону [5].

Авторы признательны Г. Е. Малашкевичу за предоставленные для численного эксперимента данные.

ЛИТЕРАТУРА

1. Озель, Ф. Е. Материалы и устройства, использующие антистоксовые люминофоры с переносом энергии / Ф. Е. Озель // ТИИЭР. – 1973. – Т. 61, № 6. – С. 87–120.

2. Лазерные фосфатные стекла / Н. Е. Алексеев [и др.] ; под. ред. М. Е. Жаботинского. – М. : Наука, 1980. – 352 с.

3. Физико-химические и спектрально-люминесцентные характеристики активированных неодимом теллуритных стекол / Г. Е. Малашкевич [и др.] // ФХС. – 1987. – Т. 13, № 6. – С. 866–873.

4. Luminescence of borogermanate glasses activated by Er^{3+} and Yb^{3+} ions / G. E. Malashkevich [et al.] // J. Non-Cryst. Solids. – 2011. – Vol. 357. – P. 67–72.

5. Малашкевич, Г. Е. К вопросу о прогнозе синтеза активированных стекол с заданными спектральными характеристиками / Г. Е. Малашкевич, Н. Н. Ермоленко – Минск, 1984. – 16 с. – (Препринт / АН БССР. Ин-т физики ; № 323).

А. А. КОЗИНСКИЙ
УО БрГТУ (г. Брест, Республика Беларусь)

МЕТОДЫ ПРЕДСТАВЛЕНИЯ ТЕКСТОВ ДЛЯ НЕЙРОСЕТЕВОЙ ОБРАБОТКИ В API KERAS

Задачи обработки естественного языка с помощью нейронных сетей включают методы, основанные на цифровом представлении текстов. Перечень таких задач

достаточно широк и включает: классификацию, генерацию и сегментацию текстов, распознавание и синтез речи и другие. Решения перечисленных задач широко используются в компьютерных переводчиках, автоматических диалоговых телеграмм-ботах, экспертных системах, антиспам средствах, голосовых помощниках и др. Несмотря на разнообразие классов задач, в большинстве случаев представление текста для нейросетевой обработки включает общие методы кодирования. Такое представление должно сокращать вычислительную сложность за счет уменьшения объема исходных данных, повышать степень их структурированности. Структурированность при этом подразумевает связь между родственными компонентами текста: словами, оборотами речи, предложениями и их последовательностями. Кроме того, основным требованием к представлению текста по-прежнему является кодирование – его числовое представление.

Выполним обзор основных методов представления текста для последующей обработки с использованием нейронных сетей. В этом случае текст преобразуется в систему обработанных (очищенных) и структурированных данных, пригодных для применения методов машинного обучения. Такая целенаправленная система, подготовленная для загрузки на вход нейронной сети, получила название «dataset» [1].

В числе основных методов представления текста назовем токенизацию. Токенизация [2] представляет собой преобразование текста в последовательность индексов слов. Индекс слова – это целочисленный идентификатор из словаря частот. Словарь частот, как правило, составляется на основе текстов достаточного объема. Для подавляющего числа задач достаточно использовать словарь объемом 20 000 слов. Однако зачастую, например, в отдельных классификациях документооборота могут быть эффективно использованы словари объемом 1000 слов. Для токенизации в API Keras (см., например, [3]) используется класс Tokenizer, что позволяет представить ресурсоемкие операции в нескольких строках кода. Токенизация в целом сохраняет представление о последовательности слов в тексте. Во фрагменте 1 представлена последовательность операций для токенизации произвольного текста (здесь и ниже код приведен «схематически» и не включает необходимые библиотеки).

Фрагмент 1. Схема использования класса Tokenizer

```
train_text  = "… токенизируемый текст может состоять из нескольких миллионов слов  "
Max_Count = 20000 # Объем словаря
filters = '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n'  # Символы для удаления из текста
tokenizer = Tokenizer(num_words=MaxCount, filters, lower=True, split=' ',
                      oov_token='unknown', char_level=False)
tokenizer.fit_on_texts(train_text)      # train_text – текст для токенизации «скармливается»
                      # экземпляру tokenizer класса Tokenizer для обучения
index_text = list(tokenizer.word_index.items()) # Итоговый dataset представленный
                      # списком индексов
```

Результаты возможной токенизации могут быть продемонстрированы выполнением кода:

```
print(items[:120]) # 120 самых частых слов
```

```
print("Размер словаря", len(items)) # Длина словаря
```

Возможный вывод:

```
[('unknown', 1), ('и', 2), ('в', 3), ('не', 4), ('я', 5), ('что', 6), ('на', 7), ,…
Размер словаря 133070
```

Опыт автора показывает, что выполнение токенизации русскоязычного текста из 2,5 миллионов слов в сервисе Colaboratory [4] требует не более трех секунд.

Для извлечения лингвистических признаков текста используется метод Bag of words (BoW) [5]. BoW также опирается на словарь, однако не учитывает порядок слов в тексте, а опирается на факт их использования. BoW-представление не дает возможности обратного восстановления текста. Однако при таком моделировании схожие по смыслу тексты имеют похожие представления, так как опираются на подобные словарные наборы.

Для представления текста в виде BoW достаточно использовать метод text_to_matrix из класса Tokenizer, на вход которого подается текст для представления в BoW:

```
# Факт присутствия (отсутствия) слова в тексте отмечается 1 (0)
# на месте соответствующего индекса вектора длиной MaxCount
xBoW = tokenizer.texts_to_matrix(texts)
print(xBoW  [0, :20])           # Начало вектора xBoW
```

Возможным выводом является строка:

```
[0. 0. 1. 0. 1. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

Длина вектора xBoW соответствует длине словаря длиной MaxCount.

Следующий метод создания dataset для текстовых данных основан на использовании Embedding-слоя [2] библиотеки Keras. Указанный слой преобразует последовательность индексов, полученных при токенизации в плотные векторы фиксированного размера. Embedding-слой используется как первый слой модели нейронной сети. Во фрагменте 2 приведена схема нейронной сети с Embedding-слоем.

Фрагмент 2. Схема нейросети со слоем Embedding

```
model = Sequential()
# Добавляем в модель Embedding-слой преобразующий
# текст длины xLen, например, равный 100 словам в вектор размерности 25
model.add(Embedding(MaxCount, 25, input_length=xLen))
model.add(SpatialDropout1D(0.2))
model.add(Flatten())
model.add(BatchNormalization())
# Другие слои нейронной сети
model.add(Dense(10, activation='softmax'))
# компиляция сети
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Рассмотренные методы представления текстов дают эффект при их разумном сочетании и, как правило, используются совместно в одной модели нейронной сети.

ЛИТЕРАТУРА

1. IBM Documentation. What is data set? [Электронный ресурс]. – Режим доступа: https://www.ibm.com/docs/en/zos-basic-skills?topic=more-what-is-data-set. – Дата доступа: 02.01.2022.

2. Using pre-trained word embeddings in a Keras model [Электронный ресурс]. – Режим доступа: https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html. – Дата доступа: 02.01.2022.

3. Русскоязычная документация Keras [Электронный ресурс]. – Режим доступа: https://ru-keras.com/home/. – Дата доступа: 02.01.2022.

4. Добро пожаловать в Colaboratory! [Электронный ресурс]. – Режим доступа: https://colab.research.google.com/. – Дата доступа: 02.01.2022.

5. A Gentle Introduction to the Bag-of-Words Model [Электронный ресурс]. – Режим доступа: https://machinelearningmastery.com/gentle-introduction-bag-words-model/. – Дата доступа: 02.01.2022.

И. А. КОЛЕСНИКОВ, А. А. ГОЛУБ

УО МГПУ им. И.П.Шамякина (г. Мозырь, Беларусь)

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА REACT NATIVE

В наше время, когда прогресс не стоит на месте, огромную роль в жизни людей занимает использование мобильных телефонов и соответствующего программного обеспечения. Так как наибольшее распространение получили устройства, работающие под управлением операционной системы Android, то в данной статье будет рассмотрен процесс создания приложения для этой системы. В большинстве случаев такие приложения создаются в IDE «Android Studio» с использованием языков программирования Java, Kotlin и соотвествующих SDK [1].

В статье будет рассмотрена разработка приложений с использованием фреймворка React Native. В качестве редактора программного кода использована программа Visual Studio Code (VS Code), которая разработана компанией Microsoft, распространяется бесплатно, имеет широкую известность, поддерживает работу с большими проектами и большинством языков программирования, имеет большое количество функций и плагинов, которые помогают разработчику в написании кода.

Фреймворк React Native предполагает использование в качестве языка программирования JavaScript. Изначально JavaScript предназначен для написания web-приложений, которые работают в браузерах Google Chrome, Yandex или Microsoft Edge, но фреймворк React Native позволяет разрабатывать приложения на языке JavaScript для мобильных устройств с операционными системами Android и iOS. [2-3].

Основные принципы работы приложений React Native практически идентичны принципам работы web-приложений React, за исключением того, что React Native управляет не браузерным DOM, а платформенными интерфейсными компонентами.