УДК 004.89
DOI: 10.15827/2311-6749.17.4.3

ПРИМЕНЕНИЕ ДОЛГОЙ КРАТКОСРОЧНОЙ ПАМЯТИ В ДИАЛОГОВОЙ МОДЕЛИ

К.И. Бубякин, goodluckhf@yandex.ru; Е.В. Пучков, к.т.н., доцент, puchkoff@i-intellect.ru
(Донской государственный технический университет, пл. Гагарина, 1, г. Ростов-на-Дону, 344000, Россия)

В статье рассмотрена генерация текста как подход, применяющийся для автоматического получения ответов на заданные вопросы. Для реализации диалоговой модели была собрана обучающая выборка и проведена ее предварительная подготовка. Описан процесс создания модели Sequence-to-Sequence, в основе которой лежит рекуррентная нейронная сеть LSTM. Модель реализована с помощью библиотеки Tensorflow. Обучение сети проводилось с применением технологии CUDA на облачном сервере Amazon. Представлены результаты работы модели на похожих диалогах.

Ключевые слова: *долгая краткосрочная память, искусственные нейронные сети, рекуррентные нейронные сети, диалоговая модель, глубокое обучение.*

В последнее время большую популярность приобрели чат-боты и интеллектуальные онлайн-консультанты. Они решают множество задач в коммуникации с людьми и уже сейчас заменяют некоторые профессии, например, банковского консультанта. Интерес к таким системам прежде всего обусловлен аппаратными возможностями и развитием интеллектуальных технологий. В связи с этим авторы статьи считают актуальной разработку интеллектуального ассистента (чат-бота) с применением диалоговой модели.

Для реализации диалоговой модели необходимо:
– собрать и подготовить данные для обучения;
– определить параметры диалоговой модели;
– реализовать диалоговую модель в виде программного кода;
– обучить модель на собранных данных;
– протестировать модель.

Сбор данных

Главным этапом в построении интеллектуального ассистента являются сбор и подготовка данных для обучения модели. Данные (диалоги) необходимо найти по тематике задачи, которая ставится перед ассистентом. Предобработка данных обычно улучшает результаты, поэтому стоит удалить конкретные названия городов, фирм и другие имена собственные. После этого предложения разбиваются на токены – части предложения (слова, знаки препинания). Далее создается словарь (ключ – значение) таких токенов и предложения переводятся в массивы ключей, что и будет являться входными данными для диалоговой модели. Был создан корпус из 890 диалогов.

Диалоговая модель

В данной работе реализована диалоговая модель Sequence-to-Sequence, предложенная компанией Google, которая успешно применяется в задаче машинного перевода. Обучение проходит с учителем, на вход подается последовательность токенов (фраза на первоначальном языке), на выходе тоже (перевод) [1]. Такую модель можно применять и для диалогов. Отличие заключается в том, что вместо перевода с одного языка на другой используется один язык. На вход подается вопрос, на выход – ответ.

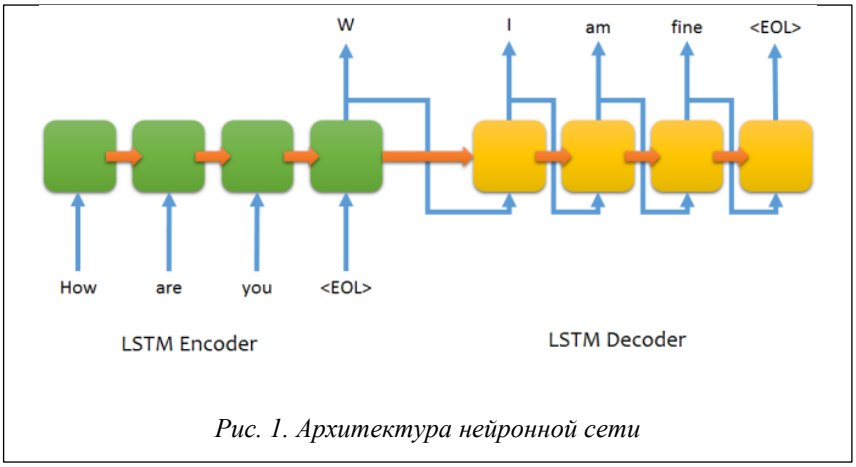

Рис. 1. Архитектура нейронной сети

В основе модели Sequence-to-Sequence лежит рекуррентная нейронная сеть LSTM [2] (рис. 1).

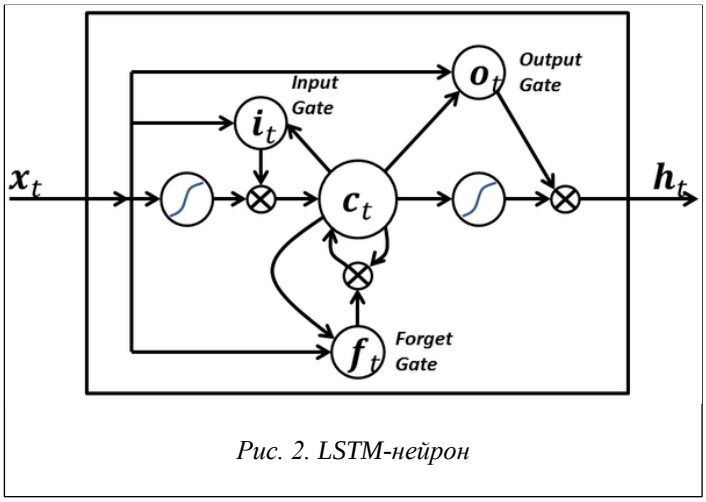

Рис. 2. LSTM-нейрон

Долгая краткосрочная память (англ. Long short-term memory; LSTM) – разновидность архитектуры рекуррентных нейронных сетей (Recurrent Neural Network, RNN), предложенная в 1997 году Сеппом Хохрайтером и Юргеном Шмидхубером [3].

Основная особенность рекуррентных сетей по сравнению с традиционными заключается в наличии обратных связей, с помощью которых сигналы, являющиеся некоторой последовательностью, с выходов этих нейронов подаются на их входы. Значения последовательности передаются вместе с предсказаниями до тех пор, пока последовательность не закончится. В отличие от простых рекуррентных сетей LSTM имеет три фильтра (входной, выходной, забывания), с помощью которых происходит управление сигналом внутри нейрона [4] (рис. 2).

Реализация модели

Для разработки была использована библиотека Tensorflow [5] компании Google. Функция, инициализирующая модель Sequence-to-Sequence, приведена ниже:

```
decoderOutputs, states = tf.contrib.legacy_seq2seq.embedding_attention_seq2seq(
    self.encoderInputs,
    self.decoderInputs,
    encoDecoCell,
    self.textData.getVocabularySize(),
    self.textData.getVocabularySize(),
    embedding_size=self.args.embeddingSize,
    output_projection=outputProjection.getWeights() if outputProjection else None,
    feed_previous=bool(self.args.test)
)
```

Код создания ячейки LSTM следующий:

```
def create_rnn_cell():
    encoDecoCell = tf.contrib.rnn.BasicLSTMCell(
        self.args.hiddenSize,
    )
    if not self.args.test:
        encoDecoCell = tf.contrib.rnn.DropoutWrapper(
            encoDecoCell,
            input_keep_prob=1.0,
            output_keep_prob=self.args.dropout
        )
    return encoDecoCell
```

self.encoderInputs – список длиной: размер серии * длина последовательности * количество нейронов на скрытом слое.

self.decoderInputs – список с такой же длиной, как и self.encoderInputs, но используется как декодер в модели sequence2sequence.

self.textData.getVocabularySize() – количество уникальных токенов.

self.args.embeddingSize – размерность каждого слова.

encoDecoCell – тип ячейки (нейрона), может быть и LSTM, и GRU.

tf.contrib.rnn.DropoutWrapper – во время обучения сети случайная часть нейронов не участвует в предсказании, что позволяет модели не переобучаться.

Параметры для создания данной модели:

– максимальная длина предложения – 20 токенов;

– количество скрытых слоев – 4;

– количество нейронов на каждом скрытом слое – 64;

– количество примеров, которые должны пройти через сеть перед обновлением весов – 32;

– количество эпох – 30.

Обучение модели

Рекуррентные нейронные сети очень требовательны к ресурсам, поэтому их обучение с использованием CPU требует очень много времени. Компания NVIDIA предложила решение данной проблемы, представив CUDA – архитектуру параллельных вычислений, позволяющую существенно увеличить вычислительную производительность благодаря использованию GPU (графических процессоров).

Компания Amazon [6] предоставляет облачный сервис, где можно арендовать видеокарту Tesla K80. С ее помощью такая модель обучалась 30 эпох в течение 15 секунд за одну эпоху (рис. 3).

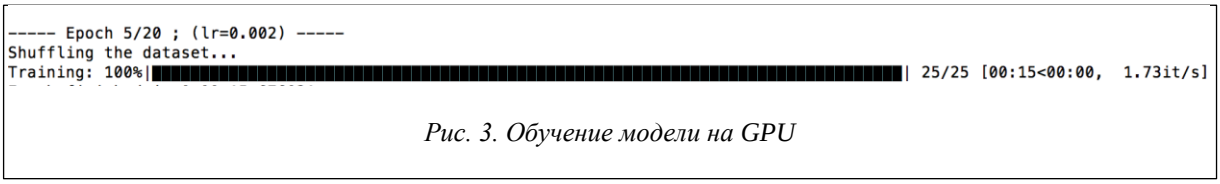

Рис. 3. Обучение модели на GPU

Для сравнения скорости вычисления модель была обучена на ноутбуке с использованием CPU. Для обучения потребовалось 2 минуты 19 секунд на одну эпоху (рис. 4).

Рис. 4. Обучение модели на CPU

Результаты

Модель была протестирована на похожих диалогах. Результаты работы модели показаны на примерах диалога, где Q – вопрос, A – ответ.

В силу того, что данных для обучения было взято не так много, модель отвечала не всегда корректно. В задаче [7] было использовано более 1 млн диалогов.

Пример 1:

```
Q: ладно удачи
A: Я справился?

Q: ну почти справились
A: Я сказал, что это твой последний шанс?

Q: ты мне угрожаешь что-ли?
A: Я думал вам рад помочь
```

Пример 2:

```
Q: какой ноутбук?
A: Так я самом деле, хватит

Q: отвечай быстро!
A: Я не выспался

Q: что ты будешь теперь делать?
A: Поздравляю, ничего ... скажите, можете ли вы перемещать курсор?
```

Пример 3:

```
Q: какой сегодня день?
A: Сегодня отличный день!

Q: почему ты так плохо отвечаешь?
A: Я стараюсь учиться

Q: ты будешь учиться?
A: Нет, я уверен, понимаете?
```

Заключение

В работе описан подход к реализации и обучению диалоговой модели с использованием сети LSTM, изложены основные принципы работы рекуррентных нейронных сетей, представлены результаты обученной модели.

Для улучшения результатов работы диалоговой модели необходимо собрать больше данных (диалогов), применить технологию word2vec для подготовки данных [8], оптимизировать параметры модели (количество слоев, количество нейронов).

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-01-00888 а.

Литература

1. Sutskever O.V., and Le Q.V. Sequence to sequence learning with neural networks. In NIPS, 2014, 1, 2, 3.
2. Greff K., Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink & Jürgen Schmidhuber (2015), LSTM: A Search Space Odyssey, arXiv:1503.04069.
3. Sepp Hochreiter (1997). Long short-term memory. Neural Computation 9 (8): 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. PMID 9377276.
4. Graves. Generating sequences with recurrent neural networks. arXiv:1308.0850, 2013. 3.
5. Dean Jeff; Monga Rajat; et al. (Nov. 9, 2015). (PDF). TensorFlow.org. Google Research.
6. Regions and Availability Zones. AWS Documentation. URL: http://docs.awsamazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html. Retrieved, 2016.
7. Auli M., Galley M., Quirk C., and Zweig G. Joint language and translation modeling with recurrent neural networks. In EMNLP, 2013.
8. Mikolov T. et al. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781.