Реализация модели генерации подмножеств естественного языка на основе продукционной модели знаний

С. Ю. Болотова, email: bolotova.svetlana@gmail.com [1]
К. В. Сиволапов, email: sivkostja@gmail.com

[1] Воронежский государственный университет

Аннотация. *В данной работе рассматриваются модель генерации осмысленных выражений естественного языка на основе продукционной модели знаний и описывается реализации системы автоматической генерации в рамках предложенного подхода.*

Ключевые слова: *генерация диалогов, N-граммы, продукционная модель, модель Seq2Seq.*

Введение

За последние несколько лет возможности компьютеров понимать человеческую речь существенно повысились. Достигнуты определенные продвижения в данной области искусственного интеллекта [1, 2]. Однако еще многие проблемы формализации естественного языка до сих пор остаются неразрешенными.

В настоящей работе обсуждается метод генерации подмножеств естественного языка, основанный на продукционной модели знаний [3]. Описана реализация программы автоматической генерации, использующая предложенный метод. Обсуждаются результаты проведенных экспериментов.

1. Модель генерации диалогов

Помочь решить проблему генерации ответов на запросы к чатам могут вопросно-ответные системы, способные принимать вопросы и отвечать на них на естественном языке. Для создания вопросно-ответных систем часто используются решения, полученные в ходе решения задачи классификации намерений пользователей. Одним из решений задачи классификации намерений может являться использование правил.

В данной работе предлагается модель генерации диалогов при помощи многомерного анализа семантических данных за счет привлечения алгоритма автоматической генерации используемых шаблонов на основе продукционных систем. Согласно этой модели, запросы группируются в категории, каждая из которых представима

© Болотова С. Ю., Сиволапов К. В., 2021

набором сходных по смыслу правил. Правила должны содержать совокупность ключевых слов, встречающихся в обрабатываемом запросе, а также возможный, отправляемый клиенту ответ, внутри которого может находиться текст, картинка или ссылка. Настраивать содержимое ответа можно при помощи особых макросов.

Продукционная система состоит из множества таких правил (иногда этот набор правил называют продукционной памятью), интерпретатора правил, который решает, когда стоит применить каждое из них, и рабочей памяти, содержащей данные, в совокупности определяющие текущий запрос. Именно структуры данных в рабочей памяти анализируются и преобразуются правилами. Обращение к правилам синхронизируется текущими данными, а интерпретатор правил управляет выбором и активизацией определенных правил.

Системы, основанные на правилах, позволяют объединять в группы, связанные фрагменты знаний (категории). Каждое продукционное правило может быть использовано независимо от других, что в свою очередь позволяет развивать базу знаний.

Данная система может быть представлена в виде совокупности категорий, которые объединяют в себе правила, используемые для ответов на сходные вопросы, приходящие со стороны клиента, а сами правила внутри категории могут быть представлены в виде дерева, где исходное правило, определяющее категорию вопроса, лежит в корне, а последующие правила являются его вершинами.

Такая система легко модифицируется при помощи добавления новых категорий и правил. Кроме того, стоит заметить, что при данном подходе возможно добавление новых правил или категорий прямо в процессе общения с клиентом, путем динамического расширения модели, однако в то же время данная модель является громоздкой и не способна учитывать формы слов и другие особенности естественного языка.

Далее будет предложен подход, использующий вероятностные модели для формирования языковых конструкций, который может быть представлен как альтернатива данному.

Для генерации ответов используется свободный набор открытых данных, содержащий записи о email людей, работавших в корпорации Enron. Стоит заметить, что выборка писем может содержать персональную информацию и данные, которые могут ее засорять. Для устранения ненужной информации используется библиотека scrubadub, которая заменяет персональные данные на заготовленные выражения. Этот пакет позволяет очищать личную информацию из данных без ущерба для конфиденциальности людей.

После очистки данных от ненужной персональной информации был получен языковой корпус, на основе которого производится генерация языковых конструкций. В контексте данной задачи используется языковая модель на основе N-грамм [4, 5]. В области обработки естественного языка N-граммы используется в основном для предугадывания на основе вероятностных моделей. N-граммная модель рассчитывает вероятность последнего слова N-граммы, если известны все предыдущие. При использовании этого подхода для моделирования языка предполагается, что появление каждого слова зависит только от предыдущих слов. В [3] была задана модель для определения вероятности употребления фразы.

Ниже рассмотрен пример с использованием фразы «Мой дядя самых честных правил», где P - является вероятностью употребления заданной фразы. Формально она задается как вероятность возникновения последовательности слов в некотором языковом корпусе (например корпусе, который был получен выше).

P = P(мой) * P(дядя | мой) * P(самых | мой дядя) * P(честных | мой дядя самых) * P(правил | мой дядя самых честных)

Таким образом можно найти вероятность того, насколько характерна данная фраза для языкового корпуса.

После определения языковой модели, фразы разбиваются на символьные последовательности произвольной длины, для обозначения границ используются разделители <start> и <end>, при этом разделители могут быть любыми, важно лишь то, что они определяют границы между последовательностями символов. Данные последовательности используются в качестве входных для задачи генерации. Стоит заметить, что выходные данные будут обусловлены входными данными, так как разбиения производятся на фразах. Данный подход связан с моделью Seq2Seq [6]. Ключевая идея подобных моделей заключается в том, что в них рассматриваются (входная последовательность) и выходные данные (выходная последовательность) как последовательности слов. Пользуясь данным подходом можно разбить фразу «Маша пошла гулять с подругой Аней» на две: <start>Маша пошла гулять с<end>,<start>подругой Аней.<end>. В контексте данной модели последовательность символов «Маша пошла гулять с» является входной, а последовательность «подругой Аней.» является выходной. На рис. 1 представлена схема генерации языковых конструкций на основе этого подхода, где <start> и <end> являются разделителями.

Модель Seq2Seq состоит из двух рекуррентных нейронных сетей (RNN): encoder (кодер), которая обрабатывает входные данные, и decoder (декодер), которая генерирует данные вывода. Для решения данной

задачи применена модификация реккурентной нейронной сети (LSTM) [5].

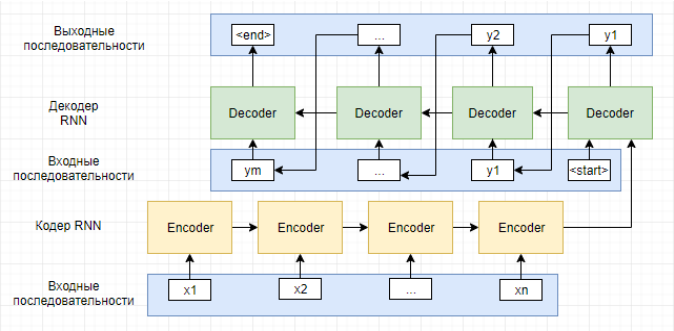

Рис. 1. Генерация выходной последовательности на основе входных данных

Схеме генерации выходных последовательностей соответствует алгоритм, состоящий из следующих основных этапов:

1) Энкодер (Encoder) последовательно считывает входное предложение и выдает вектор контекста.

2) Получив вектор, декодер генерирует последовательно продолжение фразы. Отдельными «словами» в словаре для генерация являются слова «<start>» и конец предложения «<end>».

Как было описано выше, для двух реккурентных нейросетей Encoder и Decoder используется модификация LSTM. Все рекуррентные нейронные сети имеют форму цепочки повторяющихся модулей нейронной сети. В стандартных RNN этот повторяющийся модуль будет иметь очень простую структуру, как один слой с использованием гиперболического тангенса. LSTM также имеют эту структуру, но повторяющийся модуль отличается от того, что находится в обычной реккурентной сети (рис. 2). Вместо одного слоя нейронной сети существует четыре, взаимодействующих совершенно особым образом [4].

Основным компонентом повторяющихся модулей в LSTM является состояние ячейки – горизонтальная линия, в верхней части элемента цепочки, изображенной на рис. 3. Состояние ячейки проходит напрямую через весь компонент, при этом само оно учувствует в нескольких линейных преобразованиях, что помогает не изменять элементы, проходящие через элемент цепочки, без необходимости.

Процесс удаления информации из состояния ячейки регулируется фильтрами. Фильтры пропускают входящую информацию через себя. Каждый фильтр состоит из слоя нейросети с функцией активации сигмойд и операции поточечного умножения. Сигмойд принимает значения [0,1], которое влияет на то, какую часть информации стоит пропустить через фильтр дальше (в случае 1 – всю, 0 – иначе).

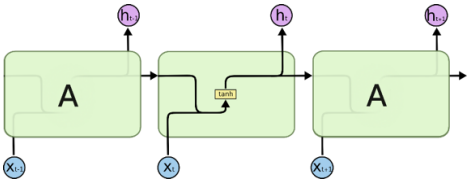

Рис. 2. Повторяющийся модуль в реккурентной нейросети

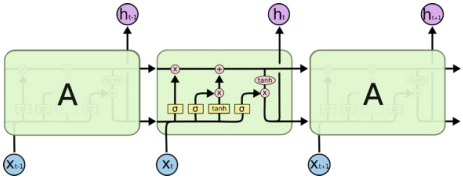

Рис. 3. Схема состояния ячейки в LSTM нейросети

2. Реализация системы

Полученная модель получила реализацию при помощи библиотеки Tensorflow с использованием надстройки Keras. Как было замечено ранее, для условной генерации использовался набор открытых данных корпорации Enron, который содержит электронные письма. Исходные данные для условной генерации изображены на рис. 4.

	from	fromn	to	ton	subject	msg
0	phillip.allen@enron.com	Phillip K Allen	tim.belden@enron.com	Tim Belden <Tim Belden/Enron@EnronXGate>		Here is our forecast
36	phillip.allen@enron.com	Phillip K Allen	muller@thedoghousemail.com	muller@thedoghousemail.com		Greg, Happy B-day. Email me your phone # and I will call you. Keith
43	phillip.allen@enron.com	Phillip K Allen	jsmith@austintx.com	jsmith@austintx.com		Jeff, What is up with Burnet? Phillip
62	phillip.allen@enron.com	Phillip K Allen	ina.rangel@enron.com	Ina Rangel		Ina, I scheduled a meeting with Jean Mrha tomorrow at 3:30
79	phillip.allen@enron.com	Phillip K Allen	bs_stone@yahoo.com	bs_stone@yahoo.com		Brenda Can you send me your address in College Station. Phillip

Рис. 4. Данные для условной генерации

Для генерации используется словарь на основе сообщений, которые были получены из дата-сета.

Код получения индекса слова и самого слова по индексу представлен в листинге 1.

Листинг 1

Получение индекса слова в словаре и получение слова по индексу

```
def create_index(self):
        for phrase in self.lang:
            self.vocab.update(phrase.split(' '))
        self.vocab = sorted(self.vocab)
        self.word2idx["<pad>"] = 0
        self.idx2word[0] = "<pad>"
        for i,word in enumerate(self.vocab):
            self.word2idx[word] = i + 1
            self.idx2word[i+1] = word
```

В качестве входных параметров для генерации выбраны n-граммы. Код формирования n-грамм приведен в листинге 2.

Листинг 2

Формирование входных и выходных n-грамм

```
processed_corpus = preprocess(corpus)
    output = []
    for line in processed_corpus:
        token_list = line
        for i in range(1, len(token_list)):
            data = []
            x_ngram = '<start> '+ token_list[:i+1] + ' <end>'
            y_ngram = '<start> '+ token_list[i+1:] + ' <end>'
            data.append(x_ngram)
            data.append(y_ngram)
            output.append(data)
    dummy_df = pd.DataFrame(output, columns=['input','output']
)
```

После обработки получаются последовательности, изображенные на рис. 5.

	input	output
159	<start> ina, i scheduled a meeting w <end>	<start> ith jean mrha tomorrow at 3:30 <end>
160	<start> ina, i scheduled a meeting wi <end>	<start> th jean mrha tomorrow at 3:30 <end>
161	<start> ina, i scheduled a meeting wit <end>	<start> h jean mrha tomorrow at 3:30 <end>
162	<start> ina, i scheduled a meeting with <end>	<start> jean mrha tomorrow at 3:30 <end>
163	<start> ina, i scheduled a meeting with <end>	<start> jean mrha tomorrow at 3:30 <end>
164	<start> ina, i scheduled a meeting with j <end>	<start> ean mrha tomorrow at 3:30 <end>
165	<start> ina, i scheduled a meeting with je <end>	<start> an mrha tomorrow at 3:30 <end>

Рис. 5. Полученные данные для загрузки в модель

Модель обучалась в 10 эпох, на каждую из которых приходилось 11 минут при обучении на графическом процессоре. В листинге 3 представлено получение модели кодера.

Листинг 3

Получение модели кодер-декодер

```
# Создание модели кодера из предыдущей модели
encoder_model = Model(encoder_inputs, [encoder_out, state_h, s
tate_c])

inf_decoder_inputs = Input(shape=(None,), name="inf_decoder_in
puts")
# Добавление переменных состояния h и c
state_input_h = Input(shape=(units*2,), name="state_input_h")
state_input_c = Input(shape=(units*2,), name="state_input_c")
decoder_res, decoder_h, decoder_c = decoder_lstm(
    decoder_emb(inf_decoder_inputs),
    initial_state=[state_input_h, state_input_c])
inf_decoder_out = decoder_d2(decoder_d1(decoder_res))
inf_model = Model(inputs=[inf_decoder_inputs, state_input_h, s
tate_input_c], outputs=[inf_decoder_out, decoder_h, decoder_c]
)
```

В листинге 4 показана функция перевода предложения к вектору, а также получение выходного предложения.

Листинг 4

Преобразование вектора в предложение

```
# Перевод предложения в вектор по индексам в языковой модели
def sentence_to_vector(sentence, lang):
    pre = sentence
    vec = np.zeros(len_input)
    sentence_list = [lang.word2idx[s] for s in pre.split(' ')]
    for i,w in enumerate(sentence_list):
        vec[i] = w
    return vec

# Функция для получения выходного предложения
def translate(input_sentence, infenc_model, infmodel):
    sv = sentence_to_vector(input_sentence, input_lang)
    sv = sv.reshape(1,len(sv))
    [emb_out, sh, sc] = infenc_model.predict(x=sv)
    i = 0
    start_vec = target_lang.word2idx["<start>"]
    stop_vec = target_lang.word2idx["<end>"]

    cur_vec = np.zeros((1,1))
    cur_vec[0,0] = start_vec
    cur_word = "<start>"
    output_sentence = ""
    while cur_word != "<end>" and i < (len_target-1):
        i += 1
        if cur_word != "<start>":
            output_sentence = output_sentence + " " + cur_word
        x_in = [cur_vec, sh, sc]
```

```
        [nvec, sh, sc] = infmodel.predict(x=x_in)
        cur_vec[0,0] = np.argmax(nvec[0,0])
        cur_word = target_lang.idx2word[np.argmax(nvec[0,0])]
    return output_sentence
```

В итоге была получена модель (рис. 6), которая по входным параметрам генерирует продолжения фраз. На рис. 7 показан пример ее использования, где на вход подается входная последовательность, а на выход - результат.

```
If using Keras pass *_constraint arguments to layers.
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #    Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 50)]              0
_________________________________________________________________________________________________
embedding (Embedding)        (None, 50, 100)           2034300    input_1[0][0]
_________________________________________________________________________________________________
input_2 (InputLayer)         [(None, None)]            0
_________________________________________________________________________________________________
bidirectional (Bidirectional (None, 50, 256), (N       235520     embedding[0][0]
_________________________________________________________________________________________________
embedding_1 (Embedding)      (None, None, 100)         2343400    input_2[0][0]
_________________________________________________________________________________________________
concatenate (Concatenate)    (None, 256)               0          bidirectional[0][1]
                                                                  bidirectional[0][3]
_________________________________________________________________________________________________
concatenate_1 (Concatenate)  (None, 256)               0          bidirectional[0][3]
                                                                  bidirectional[0][4]
_________________________________________________________________________________________________
cu_dnnlstm_1 (CuDNNLSTM)     [(None, None, 256),       366592     embedding_1[0][0]
                                                                  concatenate[0][0]
                                                                  concatenate_1[0][0]
_________________________________________________________________________________________________
dropout_1 (Dropout)          (None, None, 256)         0          cu_dnnlstm_1[0][0]
_________________________________________________________________________________________________
dense (Dense)                (None, None, 128)         32896      dropout_1[0][0]
_________________________________________________________________________________________________
dropout (Dropout)            (None, None, 128)         0          dense[0][0]
_________________________________________________________________________________________________
dense_1 (Dense)              (None, None, 23434)       3022986    dropout[0][0]
=================================================================================================
Total params: 8,035,694
Trainable params: 8,035,694
Non-trainable params: 0
_________________________________________________________________________________________________
```

Рис. 6. Полученная модель

Однако, полученные последовательности оказываются плохо связанными с ожидаемыми результатами. Чтобы исправить подобную ситуацию, можно в качестве входных и выходных n-грамм использовать не символы, а отдельные слова. Тогда входные и выходные последовательности примут вид, показанный на рис. 8.

Так как количество входных параметров уменьшилось, вместо использования символов использовались целые слова. Полученная модель обучалась в 50 эпох, среднее время обучения на графическом процессоре каждой эпохи составляло 35 секунд. На рис. 9 показаны результаты генерации для полученной модели.

	Input seq	Pred. Seq
0	hi	
1	i was out of the office on friday	for sunday's game
2	let me	you want to talk, up to make sure everything is covered. i think we should. 2 pm?
3	jeff, i am in the office today. any isssues to deal with	for my place? john
4	please can	you get a chance to dig out those names? thanks again. b.
5	thanks for	e the delay! shona
6	let me know	re: corenoncore. thx, jeff 415.782.7822
7	i would look	e to get the callin number for the project with johnnie jurewitz?
8	greg, have	the delay! shona
9	brenda can you send me	the party on sunday. your new name backout ballases!
10	here is our	attached spreadsheet for march. thanks, mark
11	i scheduled a meeting	on stages 13 based on least efficient, marginal unit. jeff

Рис. 7. Полученные результаты

	input	output
0	<start> is our <end>	<start> forecast <end>
1	<start> happy bday <end>	<start> email me your phone and i will call you keith <end>
2	<start> happy bday email <end>	<start> me your phone and i will call you keith <end>
3	<start> happy bday email me <end>	<start> your phone and i will call you keith <end>
4	<start> happy bday email me your <end>	<start> phone and i will call you keith <end>
5	<start> happy bday email me your phone <end>	<start> and i will call you keith <end>
6	<start> happy bday email me your phone and <end>	<start> i will call you keith <end>
7	<start> happy bday email me your phone and i <end>	<start> will call you keith <end>
8	<start> happy bday email me your phone and i will <end>	<start> call you keith <end>
9	<start> happy bday email me your phone and i will call <end>	<start> you keith <end>

Рис. 8. Входные и выходные последовательности для генерации ответов

	Input seq	Pred. Seq
0	hi	didn t know the first new email address thanks please cg
1	i was out of the office on friday	at 18009781788 if you have questions
2	let me	know if you need any additional data request thank you jim derrick
3	jeff i am in the office today any isssues to deal with	for your tv and lineups
4	please can	you at this stuff ruth
5	thanks for	you at 415 782 7822 thanks
6	let me know	if you have any questions or opportunities thanks martin cuilla
7	i would look	i have the distibution list for this week my america is 7138596238 thanks jean
8	greg have	a minute grab me to chat for five chris
9	here is our	email address s game
10	i scheduled a meeting	in
11	i scheduled a meeting with jean mrha tomorrow at	3 30
12	can you explain me	you get a chance i don t have class tonite
13	i want a pizza	jeff

Рис. 9. Результаты генерации для полученной модели

Заключение

Предложенная модель помогает справиться с определенным кругом задач генерации подмножеств естественного языка, а именно – автоматически анализировать запросы клиентов и генерировать персонализированные ответы на эти запросы. В результате проведенных тестов предложенная модель показала 94% верных ответов.

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00037.

Список литературы

1. Ze H. Statistical parametric speech synthesis using deep neural networks / H. Ze, A. Senior, M. Schuster // Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference. – 2013. – P. 7962–7966.

2. Exploring the limits of language modeling / Rafal Jozefowicz [et al.] // arXiv preprint arXiv:1602.02410. – 2016.

3. Болотова С.Ю. Использование продукционной модели знаний для решения задачи генерации подмножеств естественного языка / С.Ю. Болотова, К.В. Сиволапов // Материалы XX Международной научно-методической конференции «Информатика: проблемы, методы, технологии». Воронеж, 13-14 февраля 2020г. – Воронеж : «Научно-исследовательские публикации», 2020. – С. 25–30.

4. Smart Compose: Using Neural Networks to Help Write Emails. – Режим доступа: https://ai.googleblog.com/2018/05/smart-compose-using-neural-networks-to.html (дата обращения: 13.10.2019).

5. Understanding LSTM Networks – Режим доступа: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата обращения: 14.10.2019).

6. Модель seq2seq в машинном переводе – Режим доступа: https://otus.ru/nest/post/285/ (дата обращения: 14.10.2019).