**Лекция 7**

**Архитектура Sequence to Sequence**

Модель Seq2Seq состоит из двух основных блоков: encoder и decoder.



* На ячейки рекуррентной сети encoder подается исходная фраза разбитая по словам: «How are you?».
* Encoder обрабатывает её и на выходе получает некоторую закодированную последователность z.
* Decoder, помимо информации с выхода encoder-а, получает эталонный ответ на котором обучается: «I am fine».
* В процессе обучения декодер меняет свои веса таким образом, чтобы при получении исходного вопроса на вход, в идеале, выдать на выход эталонную фразу.
* При обучении фраза обрамляется стартовым с топовым тегом. В данном случае <BOS> — тег начала и <EOS> — тег окончания.

Для построения модели, которая сможет отвечать на вопросы условно будут работать две модели: тренировочная и рабочая. Сама нейронная сеть одна и та-же. Отличие только в способе использования.

**Обучение seq2seq модели**

Есть база вопросов и база ответов. Необходимо, чтобы были именно вопросы и ответы на них. Для нейронки должно быть понятно, что на текст вопроса дается определенный ответ.

Тренировочная модель в **Seq2Seq (Sequence-to-Sequence)**

Для **тренировочной модели** последовательность обучения будет следующая:

* На вход encoder подаем вопрос. Например, «Привет, как дела?».
* Encoder его закодирует, используя слой embedding для конвертации слов в многомерный вектор и LSTM.
* На выходе LSTM encoder-а возвращается состояния h и c. В коде дальше будет понятно как попросить нейронку возвращать эти два параметра.
* Декодер также содержит embedding слой для векторизации ответа и LSTM.
* Для декодера эталонный ответ из обучающей выборки обрамим тегами открытия и закрытия: <start> и <end>. Теги могут быть любыми.
* Состояния h & c с encoder-а и эталонный ответ подается на вход decoder-а. Он на нём обучается и формрует ответ. Например, «все хорошо, спасибо <end>».
* Декодер в процессе обучения «поймет», что на тег начала фразы <start> на входе и некоторому состоянию с encoder нужно начать генерировать ответ.
* Кроме того декодер «осознает», что сгенерированную последовательность он должен завершить тегом <end>.

Рабочая модель в **Seq2Seq (Sequence-to-Sequence)**

Обработка вопроса **рабочей моделью** будет следующая:

* Encoder в рабочей модели такой-же, как и в тренировочной модели. Разница лишь в том, что на его вход будет подаваться набранный пользователем вопрос, а не связка вопрос-ответ из обучающей базы.
* В рабочей модели используется ранее обученный декодер, но на вход ему будет подан только тег <start>.
* Декодер «понимает», что по приходу тега <start> нужно взять состояние с encoder и сгенерировать какое-то (одно) слово ответа.
* В идеале он сгенерирует первое слово в последовательности: «всё».
* Полученное слово «всё» подается на вход декодера вместе с состоянием полученным на предыдущем шаге на выходе декодера.
* Затем полученное слово «всё» вновь подается на вход декодера совместно с состоянием полученным на предыдущем шаге. На выходе декодер формрует слово <хорошо>.
* Новое слово вместе с состоянием в цикле вновь подается на вход декодера до тех пор, пока декодер не решит, что фраза завершена и вернет тег <end>.

**Тренировочная модель**

**Работа encoder-а seq2seq**

Рассмотрю по шагам работу encoder-а.

|  |  |
| --- | --- |
| На входе фраза очищенная от знаков пунктуации:  | [Привет как дела]  |
| Предложение подается на Tokenizer Keras для преобразования в последовательность чисел. На выходе 3 числа по количеству слов на входе.  | [95 18 10]  |
| Длина вопроса может быть разной, а размерность входа нейронки фиксированная. Нужно все вопросы привести к одной длине. Это делается добавлением 0-ей. Например, длина вопроса может быть не более 5 слов. Дополняем наши 3 цифры 2-мя нулями  | [95 18 10 0 0]  |
| На входе encoder-а первым стоит слой embedding.  |  |
| Слой **embedding** преобразует каждое слово в векторное пространство с заданной нами размерностью. Например, первым идет слово привет закодированное числом 95. Слой embedding преобразует это слово, например, в 200 мерное пространство.  |  |
| Слой **embedding** обучается также back propagation-ом, чтобы получить заданное нами n-мерное пространство из исходного слова.  |  |
| Полученная матрица будет подана на LSTM.  |  |

| **Encoder (слои + результаты)** |  |  |
| --- | --- | --- |
| Embedding  |  |  |
| LSTM |  |  |
| На выходе LSTM — encoder state (ES): | h | c |

**Работа decoder-а seq2seq**

|  |  |
| --- | --- |
| Берем ответ на заданный вопрос из обучающей выборки.  | [Спасибо все хорошо]  |
| На первом шаге добавляем к этой последовательности теги начала <start> и конца <end> | [<start> Спасибо все хорошо <end>]  |
| Предложение подается на Tokenizer Keras для преобразования в последовательность чисел. На выходе 5 чисел по количеству слов на входе.  | [1 45 18 24 2]  |
| Длина вопроса может быть разной. Вопросы приводим к одной длине добавлением 0-ей в последовательность слов. Например, максимальная длина ответа может быть равна 7 — добавляем в конец два 0-я. | [1 45 18 24 2 0 0]  |

| **Decoder (слои + результаты и доп. входы)** |  |  |
| --- | --- | --- |
| Embedding  |  |  |
| С encoder-а ES на вход -> | LSTM  |  |
| Значения со всех ячеек LSTM | h | c |
| Dense c кол-ом нейронов = длине словаря и функцией активации — softmax |  |  |