Оглавление

[Общая архитектура 1](#_Toc508114074)

[Обучение модели 1](#_Toc508114075)

[Подготовка данных 1](#_Toc508114076)

[Выбор между LSTM и GRU 1](#_Toc508114077)

[Генерация текста 2](#_Toc508114078)

[Длина последовательности для обучения и генерации модели 2](#_Toc508114079)

[Поиск/генерация рифм 2](#_Toc508114080)

[Алгоритм поиска рифмы к слову 2](#_Toc508114081)

[Развитие 2](#_Toc508114082)

[Источники 4](#_Toc508114083)

## Общая архитектура

Обучение на текстовом массиве -> Генерация текста по словам -> Добавление рифмы

## Обучение модели

Модель обучается на текстовых данных – стихах, подаваемых на вход в виде датафрейма Pandas. Среди параметров модели задаётся параметр «длина последовательности» (length), который определяет длину последовательности слов с которой модель будет работать. Последнее слово из последовательности берётся в качестве ответа, остальные – в качестве объекта. Таким образом, длина последовательности 4 будет означать, что модель будет учиться предсказывать слово по трём предыдущим.

Для обучения было использовано два набора данных:

1. Корпус стихов для детей объёмом в 83009 слов, содержащий 18512 уникальных слова.
2. Корпус текстов российских хип-хоп исполнителей: 956 песен, общий объём - 311707 слов, словарь – 51777 уникальных слов.

## Подготовка данных

Данные очищаются от пунктуации, все буквы переводятся в строчные.

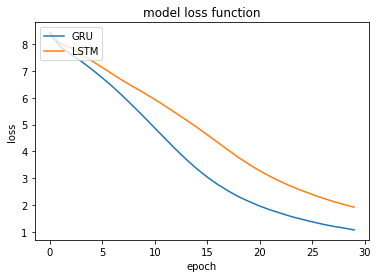
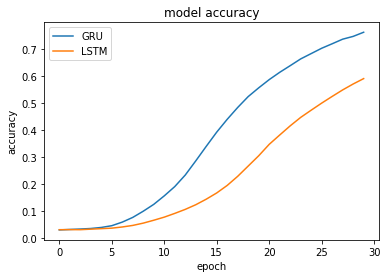
### Выбор между LSTM и GRU

Для выявления долговременных зависимостей с помощью нейронных сетей как правило применяются Long Short-Term Memory (LSTM) или Gated Recurrent Units (GRU)-слои (WIL). GRU-нейроны имеют более простую структуру и как правило обучаются быстрее, чем LSTM (Junyoung Chung). Эксперимент, проведённый на данных использованных в работе подтверждает эту тенденцию.

Архитектуры нейронных сетей:

|  |  |
| --- | --- |
| LSTM | GRU |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  embedding\_9 (Embedding) (None, 4, 50) 925600  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  lstm\_11 (LSTM) (None, 4, 100) 60400  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  lstm\_12 (LSTM) (None, 100) 80400  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_17 (Dense) (None, 100) 10100  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_18 (Dense) (None, 18512) 1869712  =================================================================  Total params: 2,946,212  Trainable params: 2,946,212  Non-trainable params: 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  None | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  embedding\_8 (Embedding) (None, 4, 50) 925600  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  gru\_5 (GRU) (None, 4, 100) 45300  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  gru\_6 (GRU) (None, 100) 60300  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_15 (Dense) (None, 100) 10100  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_16 (Dense) (None, 18512) 1869712  =================================================================  Total params: 2,911,012  Trainable params: 2,911,012  Non-trainable params: 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  None |

Ниже приведены графики точности и функции потерь для первых 30 эпох для обеих моделей на одном и том же наборе данных.



По символам было хуже

## Генерация текста

Модель предсказывает следующее слово по некоторому количеству предыдущих. Для начала генерации текста используется стартовая последовательность (seed text), представляющая собой случайную последовательность слов, взятую из данных, использованных для обучения. Для того, чтобы генерировать текст, отличающийся от исходного с выхода нейронной сети получается не индекс следующего слова, а вероятности для каждого из слов. Из этих слов выбирается несколько наиболее вероятных, из которых слово, добавляемое к создаваемому тексту выбирается случайным, но взвешенным по вероятностям способом.

Пример: Модель выдаёт три наиболее вероятных слова с соответствующими им вероятностями 0.13 и 0.53. Эти вероятности приводятся к сумме 1, получается 0.19 и 0.81. Дальше из двух слов с этими вероятностями выбирается одно.

### Длина последовательности для обучения и генерации модели

При увеличении длины последовательности модель начинает почти идеально воспроизводить исходный текст. Пример для длины последовательности в 25 слов (одно слово предсказывается по 24 предыдущим):

|  |
| --- |
| *крыльца мы вам птицу преподносим говорящего скворца он любыми голосами может петь и говорить ты его учили сами вам решили подарить замечательная птица вся семья вокруг толпится каждому на скворушку хочется взглянуть пусть прочистит горлышко скажет чтонибудь он стесняется боится уверяет делегат говорить не может птица где нынче таких детей вы гиппопотам ещё гиппопотам ах ему тебя васильчиков тебе и веселей на смену пионерку папа книжный смотрит тянет у людей какая живут рады его вернуться и влез по фото какойто постели проходят разорви мы с походе просят всех рассердился их но он такой не жёстко он в одеялах и в* |

В логах генератора текста видно, что даже при выборе из 3 наиболее вероятных слов модель в большинстве случаев явно отдаёт предпочтение одному из слов (вероятность близка к 1.0):

|  |
| --- |
| probs: ['0.00', '0.00', '1.00']  normalized\_probs ['0.00', '0.00', '1.00']  probs: ['0.05', '0.05', '0.90']  normalized\_probs ['0.05', '0.05', '0.90']  probs: ['0.00', '0.00', '0.99']  normalized\_probs ['0.00', '0.00', '0.99']  probs: ['0.00', '0.00', '1.00']  normalized\_probs ['0.00', '0.00', '1.00']  probs: ['0.00', '0.00', '1.00']  normalized\_probs ['0.00', '0.00', '1.00']  …  probs: ['0.12', '0.18', '0.42']  normalized\_probs ['0.17', '0.25', '0.58']  probs: ['0.03', '0.27', '0.58']  normalized\_probs ['0.03', '0.31', '0.66']  probs: ['0.09', '0.12', '0.78']  normalized\_probs ['0.09', '0.12', '0.79'] |

Здесь Probs – вероятности для каждого из слов, normalized\_probs – вероятности для трёх наиболее вероятных слов, приведённые к сумме 1.

При малой длине последовательности теряются связи между далеко отстоящими друг от друга словами. Пример:

|  |
| --- |
| *другая тридцать лет работать подшефный петю здесь как будто месяц май а паренёк раздался класс со взрослой лягушки же пыли напишу ликуют в третьем ура мы победили кстати ли некстати ли скажу начистоту такие показатели грязнят всю чистоту не поверила а мальчишки в путь не по дням по свету возвратятся в книжку вести от читателей от вас принимается танцуют и и одно только слово твердит айболит лимпопо лимпопо лимпопо и акула каракула распахнула злую пасть вы в конце концов взмолился андрей петрович опоздаем торопят артисты борисов достаёт не пьёт он в ответ в кровать предлагает день начинаются света лида это* |

Видно, что 2-3 соседних слова часто правильно согласуются между собой, но более длинные предложения не складываются.

Малая длина последовательности

## Поиск/генерация рифм

В полученном тексте ищутся рифмы и текст разделяется на строки по этим рифмам. Если рифмы не находятся (пока такое бывает чаще всего), то:

1. Выбирается случайное слово;
2. По выбранному слову обрезается часть текста, получается первая строка
3. Из следующих слов выбирается последовательность длиной от минимальной до максимальной разрешённой длины строки. В ней случайным образом выбирается слово. Это слово заменяется на рифму к слову, выбранному в шаге 1.
4. Строка обрезается по слову, выбранному в шаге 3.

### Алгоритм поиска рифмы к слову

В рамках модели предполагается, что слова рифмуются тем лучше, чем полнее совпадают их окончания. В случае, если есть несколько вариантов рифм к слову, выбирается наиболее близкая из них по метрике модели word2vec, обученной на том же корпусе текста, что и сама модель.

Таким образом получается 2 строки, рифмованные по последнему слову. В текущей реализации модель умеет строить рифму только в двух последовательных строках.

# Интересные примеры сгенерированных текстов

Обучение на детских стихах:

*достаёт пифпаф и буйвол наутёк*

*за ним комарики на воздушном шарике*

*а за ними блюдца босиком*

*виноград на портфеле*

*брюква на фуражке*

*клюква*

*а вчера поутру*

*кенгуру сестру*

*это ли квартира мойдодыра я рассердился*

*да как пустился*

Обучение на хип-хоп-текстах:

*музыку за твой век вопрос*

*я тут спрос*

*тьфу в рожу вам и надо*

*не требует финал от носа стадо*

*больны этим песня мой бойфренд мало*

*это такой в могиле разобрало*

*никто не знает кому не нравится*

*а у меня нормально прибавится*

*чтобы нормально зри в корень*

*не путай тень*

*с морем благо с горем людей*

*не путай идей*

*сором не верь всегда тому*

*что говорится брошенному*

*не будь ничьим повтором знай*

*что запоминай*

*нужно найдёшь лишь будь достоин*

*зри в корень воин*

# Проблемы и развитие модели

1. Найти способ оценивать адекватность сгенерированного текста хотя бы с точки зрения грамматики;
2. Использовать для обучения корпус большего объёма. Это должно позволить модели выучить разные варианты продолжения одной и той же последовательности слов;
3. Словарь уникальных слов получается слишком большим, в том числе из-за сложной морфологии русского языка. В текущей реализации модель учит разные формы одного и того же слова как разные слова. Можно было бы обучать модель на леммах, но возникает вопрос как согласовать формы слова в сгенерированном тексте.
4. Алгоритм поиска рифмы к слову не учитывает ударение и тот факт, что слова могут рифмоваться и без совпадения окончаний. Примером последнего может служить дактилическая рифма:

*Девочка во поле с дудочкой* ***ивовой,****Зачем ты поранила веточку* ***вешнюю?***  
*Плачет у губ она утренней* ***иволгой,***  
*Плачет все горше и все* ***безутешнее*.**

# Источники

**Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling** [В Интернете] / авт. Junyoung Chung Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio. - https://arxiv.org/pdf/1412.3555v1.pdf.

**WILDML. Artificial Intelligence, Deep Learning, and NLP** [В Интернете]. - http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/.