

# Seq2Seq 生成式对话

郭帅帅 3160104060//何洪良 3160103176

June 2019

## 1 Introduction

聊天机器人是目前自然语言处理中非常火热的一块内容，虽然因为一些技术的原因，目前的效果并不算理想，并且热度有所下降，但其未来的发展依然是富有前景。Seq2Seq 则是现在的聊天机器人中最普遍使用的方式。我们小组将采用该方法体验聊天机器人的开发过程并实现一定的效果。

Seq2Seq 是一种**端到端的序列学习**方法，是 RNN 的一种高级复合结构，适合于输入和目标都是**不定长序列**的场景，如机器翻译、文本生成、时间序列分析、语言概率模型等。其基本思想是：通过一个 RNN 将输入序列“编码”为一个固定大小的状态向量  $h$ ，也叫“隐状态”；再通过第二个 RNN 将状态向量“解码”为输出序列。这里第一个 RNN 通常被称为 Encoder，第二个 RNN 被称为 Decoder，而且这里的 RNN 一般用 LSTM 或 GRU 来实现。

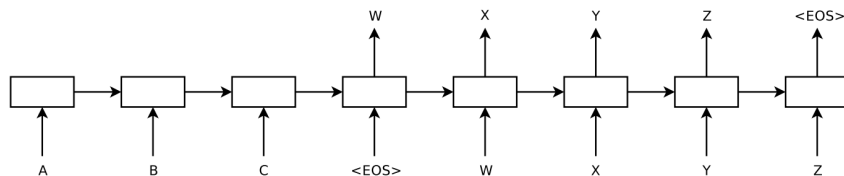
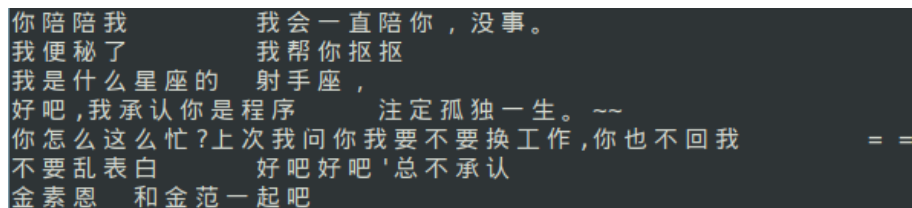


图 1: Seq2Seq 架构 [1]

## 1.1 数据集

这里的数据使用了开源的小黄鸡语料和 *ChatterBot* 中文语料 [2]。其中，小黄鸡语料有对话 40 万余条，*ChatterBot* 仅有 600 多条，但质量较高。所有语料随机取示例对话有：



```
你陪陪我      我会一直陪你，没事。
我便秘了      我帮你抠抠
我是什么星座的 射手座，
好吧，我承认你是程序    注定孤独一生。~~
你怎么这么忙？上次我问你我不要换工作，你也不回我    = =
不要乱表白      好吧好吧'总不承认
金素恩 和金范一起吧
```

图 2: 对话示例样本

这是已经处理好的数据，每行为一个对话，其结构为“问 + Tab + 答”。

## 1.2 代码实现

最初我们按照 PyTorch 官方教程的 AI-Chatbot[3] 搭建了一个版本，但效果较差。后来用 Keras 重写了一遍，将代码大大简化，并且达到了令人满意的效果。其基本结构如图 3 所示。

在训练阶段，本系统使用 256 维的隐变量 ( $h \in \mathbb{R}^{256}$ )，优化器为 *RMSprop*，学习率为 0.01，使用 *Categorical CrossEntropy* 作为 loss。

在预测阶段，本系统使用 Beam Search 算法 [4] 来提高搜索最优解的性能。最常用的 Greedy Search 仅仅在每一步（每次生成一个字符）时选择概率最大的，但这一般不会等于全局最优解（联合概率最大），因此这是一种启发式搜索，只能近似最优解。而 Beam Search 则始终保留概率最大的 K 个分支，在这些最有可能分支上往下递归搜索。所以 Greedy Search 实际上是 Beam Search 在 K=1 时的情况，一般 K 越大，搜索结果越接近全局最优解，相应的搜索速度也会降低。这里取 K=12，在性能和速度上达到了一个平衡。

由于 Google Colab 的硬件性能和训练时间有限，我们仅仅使用了不到四十分之一的数据量，即一万条对话，分为多次训练。最终训练误差降低到 0.003（如图 4 所示）。

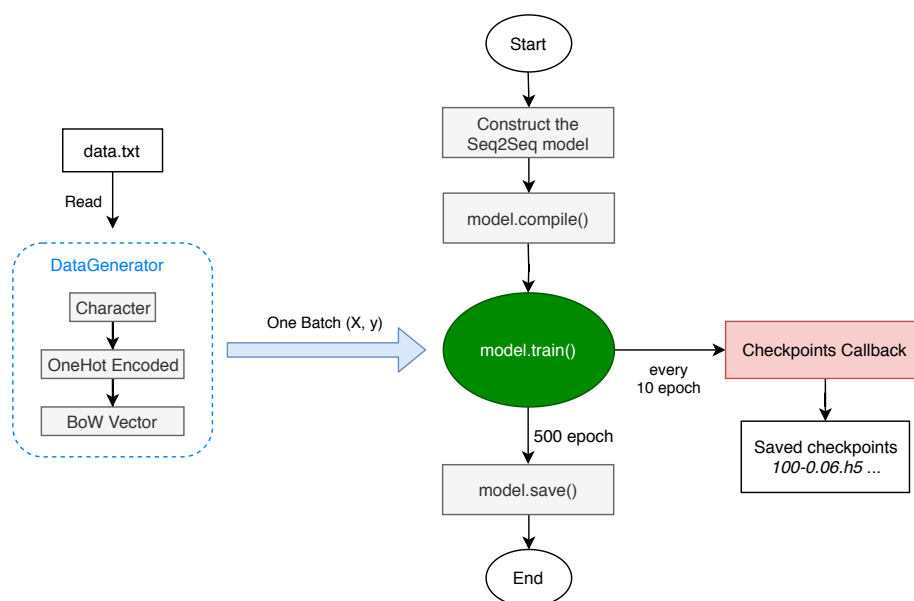


图 3: 训练算法流程

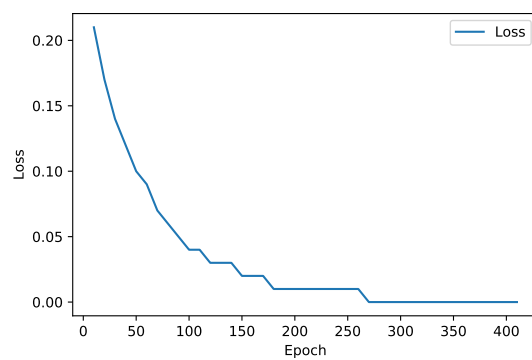


图 4: 训练误差下降曲线

人工输入对话的测试结果如图 5所示。

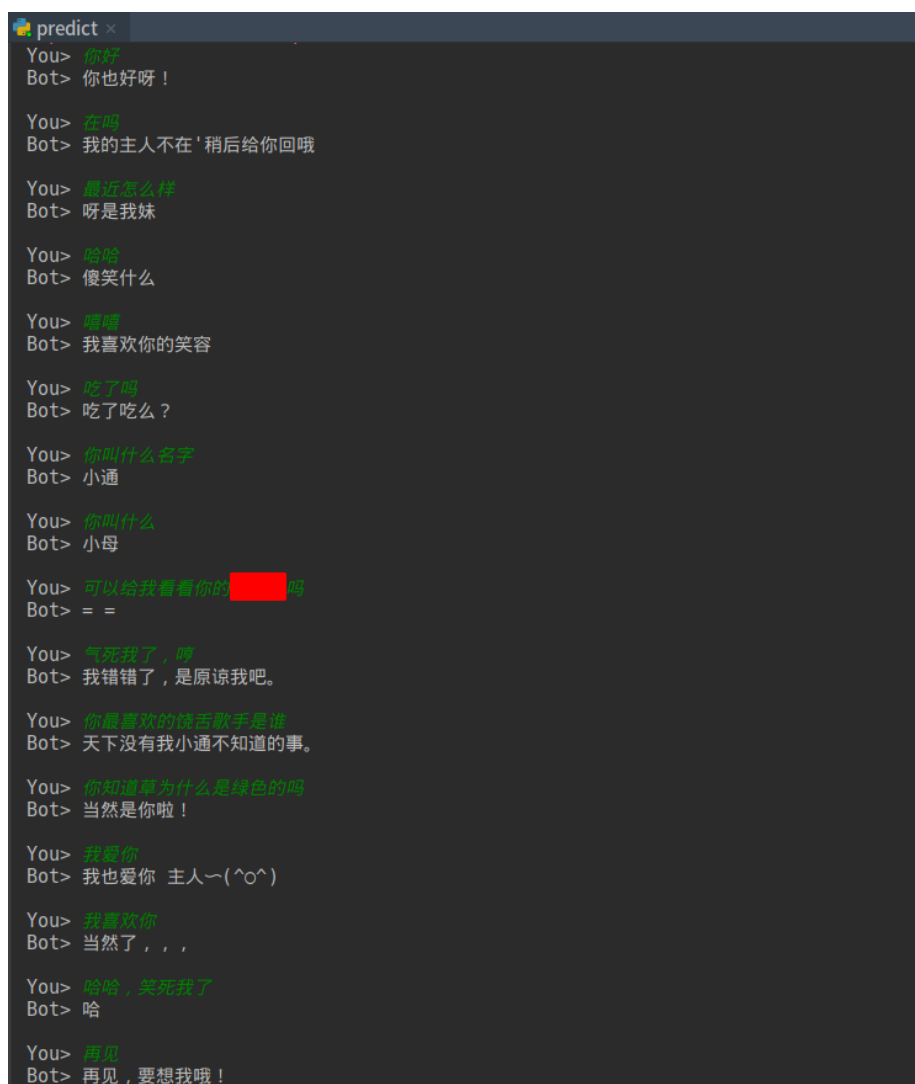


图 5: Seq2Seq 人工输入效果

## 2 Web App 前端

前端使用 Vue.js 搭建网页前端，Flask 作为中间 API 服务器与后端程序交互。可以方便部署到服务器，并且具有跨平台运行能力。前端的最终效果如图 6 所示。



图 6: 最终前端效果

## 2.1 运行步骤

本项目已经将全部依赖打包成了 Docker 镜像，使用 Docker 可以直接运行本项目，方法如下：

### 1. 下载 Docker 镜像 从以下链接下载该镜像（570MB）：

<https://pan.zju.edu.cn/share/609e33d211bb4df44a6c3c5bd8>

2. **从压缩包恢复镜像** 执行下面的命令将镜像读入本地:

```
$ docker load < s2s_image.tar.gz
```

3. **从镜像运行容器** 执行下面的命令, 从镜像运行一个容器示例:

```
$ docker run -it --rm -p 8080:8080 -p 5000:5000 s2s:v0.2 /run.sh
```

4. **打开本机浏览器开始使用** 打开浏览器, 输入地址 127.0.0.1:8080 开始使用聊天界面。

### 3 未来展望

在早期的 pytorch 版本中虽然有了 attention 但是数据集噪音太多, 后期的 keras 未采用 attention 以致于对长对话的处理能力不够。传统编码器-解码器结构在编解码时都依赖于内部一个固定长度向量。当输入序列非常长的时候, 我们难以学到或者说编码这样一个固定的向量。

因此第一个后期展望是引入 attention 机制。保留 LSTM 编码器对输入序列的中间输出结果, 对这些输入进行选择性的学习。后期我们的机器人在日常短对话的处理效果上, 还是十分令人满意的, 在长对话的处理方面希望进一步加强。

尽我们最大的努力, 实现一款能真正投入应用的萌萌哒小黄鸡聊天机器人。

### 参考文献

- [1] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- [2] “candlewill/dialog\_corpus: 用于训练中英文对话系统的语料库 datasets for training chatbot system,” [https://github.com/candlewill/Dialog\\\_Corpus](https://github.com/candlewill/Dialog\_Corpus), (Accessed on 06/12/2019).
- [3] M. Inkawhich, “Chatbot tutorial —pytorch tutorials 1.1.0 documentation,” [https://pytorch.org/tutorials/beginner/chatbot\\\_tutorial.html](https://pytorch.org/tutorials/beginner/chatbot\_tutorial.html), (Accessed on 06/12/2019).

- [4] J. Brownlee, “How to implement a beam search decoder for natural language processing,” <https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/>, Jan 2018, (Accessed on 06/12/2019).