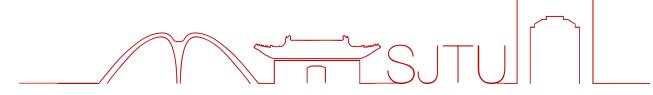




上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



AI-Poet

一款基于深度学习的 AI 写诗系统

徐惠东

2021年12月30日

饮水思源 · 爱国荣校



目录

-
- A large, semi-transparent background image of a traditional Chinese building with intricate carvings and colorful decorations, such as red roofs with blue and gold accents and white stone carvings. The image serves as a backdrop for the numbered sections.
- 1 简介
 - 2 代码实现
 - 3 运行展示
 - 4 项目总结



简介

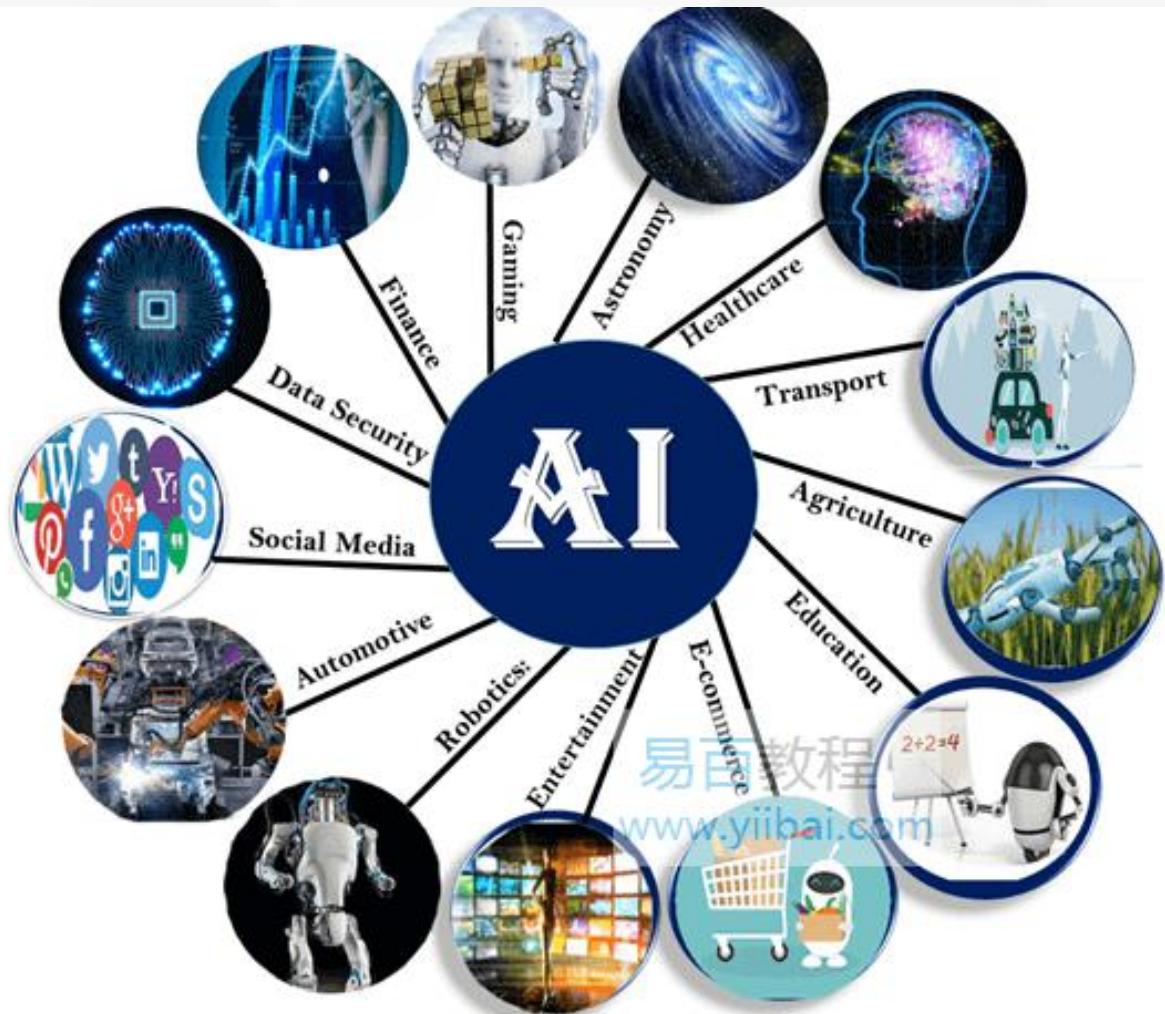
01

AI-Poet：一款基于深度学习的 AI 作诗系统





人工智能广泛应用



人工智能已经渗透了我们生活的方方面面，给我们的衣食住行都带了许多便利。

然而，在满足了人们的基本生活需求之后，精神上的追求将逐步凸显。

AI-Poet 即是一款基于深度学习的 AI 作诗系统。



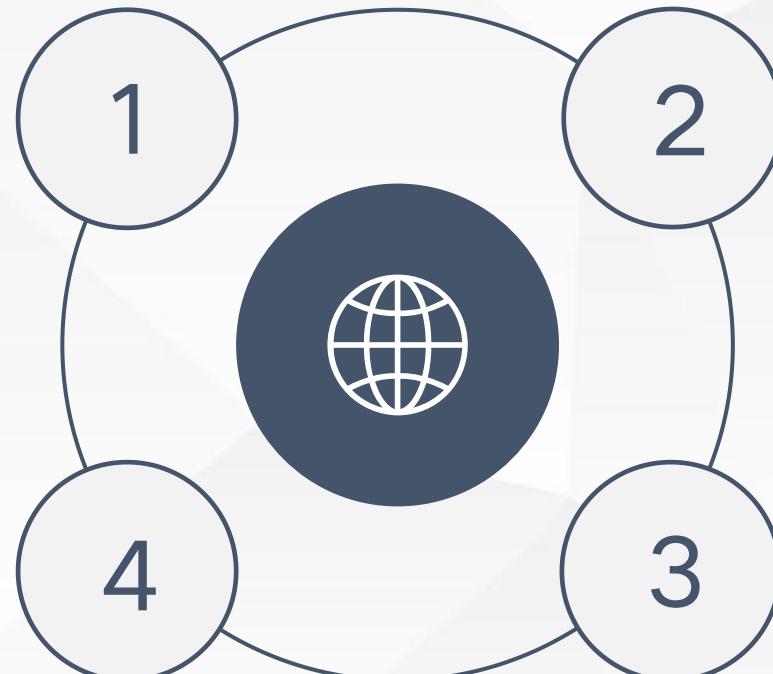


深度学习

深度学习是机器学习中一种基于对数据进行表征学习的算法，能够用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。

预训练模型

使用尽可能多的训练数据，从中提取出尽可能多的共性特征，从而能让模型对特定任务的学习负担变轻。之后还可以进行微调，可拓展性高。



循环神经网络 (RNN)

循环神经网络是一类以序列数据为输入，在序列的演进方向进行递归且所有节点（循环单元）按链式连接的递归神经网络，在对序列的非线性特征进行学习时具有一定优势

长短期记忆 (LSTM)

特殊的循环神经网络。适合于处理和预测时间序列中间隔和延迟非常长的重要事件。

02

代码实现

Notebook, Google, Colab, Google Drive





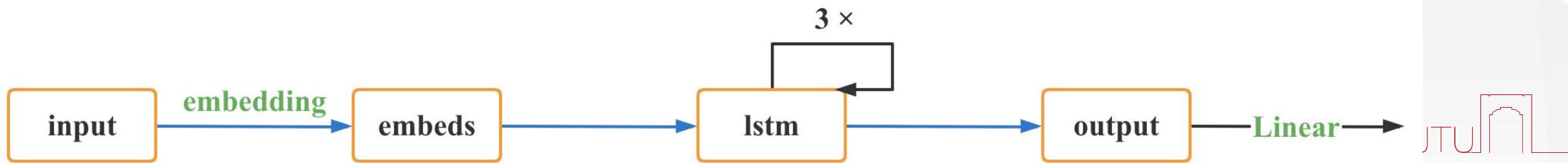
代码实现 -- 接口

```
class Config(object):
    num_layers = 3 # LSTM层数
    ...
    lr = 1e-3
    weight_decay = 1e-4
    use_gpu = True
    epoch = 30
    batch_size = 25
    maxlen = 125 # 超过这个长度的之后字被丢弃，小于这个长度的在前面补空格
    plot_every = 200 # 每20个batch 可视化一次
    max_gen_len = 200 # 生成诗歌最长长度
    ...
    embedding_dim = 256
    hidden_dim = 512
    ...
```

LSTM 层数：在大规模翻译任务的经验中，简单的堆叠 LSTM 层最多可以工作 4 层，很少工作 6 层，超过8层就很差了。本模型选择使用 3 层。

epoch 参数：向前和向后传播中所有批次的单次训练迭代次数。可以调参。

batch_size 参数：基于梯度下降，每次训练使用样本数量。太小容易欠拟合。





代码实现 -- 模型定义

```
class PoetryModel(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim):
        super(PoetryModel, self).__init__()
        self.hidden_dim = hidden_dim
        # 词向量层, 词表大小 * 向量维度
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        # 网络主要结构
        self.lstm = nn.LSTM(embedding_dim, self.hidden_dim, num_layers=Config.num_layers)
        # 进行分类
        self.linear = nn.Linear(self.hidden_dim, vocab_size)

    def forward(self, input, hidden=None):
        seq_len, batch_size = input.size()
        # print(input.shape)
        if hidden is None:
            h_0 = input.data.new(Config.num_layers, batch_size, self.hidden_dim).fill_(0).float()
            c_0 = input.data.new(Config.num_layers, batch_size, self.hidden_dim).fill_(0).float()
        else:
            h_0, c_0 = hidden
        # 输入 序列长度 * batch(每个汉字是一个数字下标),
        # 输出 序列长度 * batch * 向量维度
        embeds = self.embeddings(input)
        # 输出hidden的大小: 序列长度 * batch * hidden_dim
        output, hidden = self.lstm(embeds, (h_0, c_0))
        output = self.linear(output.view(seq_len * batch_size, -1))
        return output, hidden
```





代码实现 -- 训练过程

```
loss_data = []
# 进行训练并画图
def train():
    f = open(Config.result_path, 'w')
    for epoch in range(basic_start, Config.epoch):
        loss_meter.reset()
        for li, data_ in tqdm.tqdm(enumerate(dataloader)):
            # 将数据转置并复制一份
            data_ = data_.long().transpose(1, 0).contiguous()
            # 注意这里，也转移到了计算设备上
            data_ = data_.to(device)
            Configimizer.zero_grad()
            # n个句子，前n-1句作为inout，后n-1句作为label，二者一一对应
            # 经过 LSTM 网络进行前向传播
            input_, target = data_[:-1, :], data_[1:, :]
            output, _ = model(input_)
            # 误差反向传播
            loss = criterion(output, target.view(-1))
            loss.backward()
            Configimizer.step()
            loss_meter.add(loss.item())
            # 存储 loss 数据，方便之后画图
            loss_data.append(loss)
            # 进行可视化
            if (1 + li) % Config.plot_every == 0:
                print("训练损失为%s\n" % (str(loss_meter.mean)))
            ...
train()
```





代码实现 -- 诗句生成 -- 首句生成

```
# 给定首句生成诗歌
def generate(model, start_words, ix2word, word2ix, prefix_words=None):
    ...
    # 若有风格前缀，则先用风格前缀生成hidden
    if prefix_words:
        for word in prefix_words:
            output, hidden = model(input, hidden)
            input = input.data.new([word2ix[word]]).view(1, 1)

    # 开始真正生成诗句，如果没有使用风格前缀，则hidden = None, input = <START>
    # 否则，input就是风格前缀的最后一个词语，hidden也是生成出来的
    for i in range(Config.max_gen_len):
        output, hidden = model(input, hidden)
        if i < start_words_len:
            ...
            input = input.data.new([word2ix[w]]).view(1, 1)
        else:
            ...
            input = input.data.new([top_index]).view(1, 1)
        if w == '<EOP>':
            del results[-1]
            break
    return results
```

优先使用风格前缀生成隐藏层，并结合用户输入的首句喂给预训练模型生成下一句。
再使用生成的下一句作为下一次迭代的输入，不断迭代直至达到最大生成字数或遇到终止符 <EOP> 为止。





代码实现 -- 诗句生成 -- 藏头诗

```
# 给定首句生成诗歌
def generate(model, start_words, ix2word, word2ix, prefix_words=None):
    ...
    # 若有风格前缀，则先用风格前缀生成hidden
    if prefix_words:
        for word in prefix_words:
            output, hidden = model(input, hidden)
            input = input.data.new([word2ix[word]]).view(1, 1)

    # 开始真正生成诗句，如果没有使用风格前缀，则hidden = None, input = <START>
    # 否则，input就是风格前缀的最后一个词语，hidden也是生成出来的
    for i in range(Config.max_gen_len):
        output, hidden = model(input, hidden)
        if i < start_words_len:
            ...
            input = input.data.new([word2ix[w]]).view(1, 1)
        else:
            ...
            input = input.data.new([top_index]).view(1, 1)
        if w == '<EOP>':
            del results[-1]
            break
    return results
```

优先使用风格前缀生成隐藏层，并结合用户输入每次喂给模型一个字作为开头并续写，迭代更新至用户输入用完为止。



03

运行展示

1. 首句生成
2. 藏头诗
3. 风格选择





运行展示 - 首句生成

草色青青柳色黄，
紫梨花谢春风开。
江水东流无社稷，
君王朝日如冬霜。
白头老少今何幸，
每见知君心不伤。

寂寞梧桐深院锁清秋，
灯火暗悠悠。
宛转芳兰满，芊绵坠露生。
露禽啼不寐，惊鸟不闻声。
悄悄星河晓，团团月殿横。

明月几时有，秋风人未归。
青春来取道，春日向前飞。
洛阳桃李红，泪尽湘水流。
一杯须更醉，一日无所求。
朝朝海上起，细发斗中愁。
乡思日浩渺，妾思烟水流。
离怀一水色，何处不堪愁。
一醉不可识，日暮水东流。

JJ

JJ

JJ





运行展示 -- 藏头诗



爱君古贤者，饮造古太平。
国有圣贤子，水有帝王城。
荣名贵相府，思国势不平。
校奉两仪血，源厉万姓名。

JJ

我有一人承晓镜，
喜君发我与君恩。
欢娱未得知君意，
你竟无言亦不还。

JJ





运行展示 - 风格

不使用风格



红藕香残玉簟秋，鸳鸯一卷掩玲旌。
一声清晓起秋月，万籁千声惹九秋。



春风得意马蹄疾，一日看尽长安花。



红藕香残玉簟秋，一旦春风携去来。
长歌宛转怨不见，一曲一曲歌声来。



使用首句风格



红藕香残玉簟秋，水精帘暖魂相续。
江南昔日不得游，落日孤舟漾楚波。



八百里分麾下炙，五十弦翻塞外声



红藕香残玉簟秋，风吹雨洒江楼席。
忆昔湖畔选旅人，今年战士归江上。



04

项目总结

1. 项目进展
2. 创新点
3. 个人收获 & 致谢





项目总结 -- 项目进展

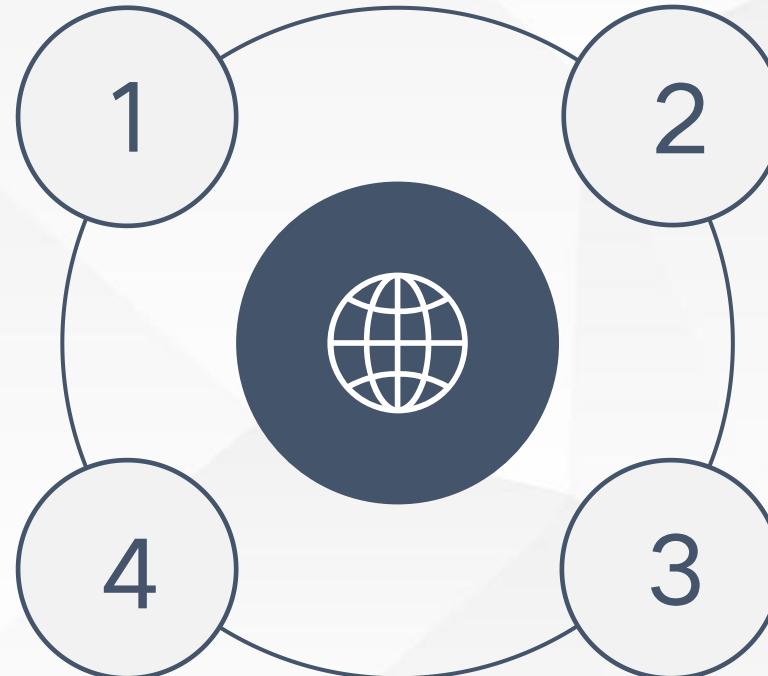


2021.11.25

正式开题，组队未遂，于是决定自己独立成队，这样可以大大拓宽我选题和做题的灵活性。

2021.12.18

模型第一次训练成功，得到相应预训练模型，根据这次预训练模型进行人工测试并且调整超参数重新训练。



2021.12.10

决定运用深度学习训练诗歌模型，开始搜索和整理网络资料，并配置模型训练环境。

2021.12.28

完成模型训练和测试，完成所有报告材料，准备答辩。





项目总结 -- 创新点





项目总结 -- 个人收获 & 致谢



个人收获

在一个学期的《人工智能》选修课中，我学习到了诸如机器学习、深度学习、自然语言处理等多种人工智能相关技术，也逐步了解到自动驾驶、智能推荐等人工智能的广大应用前景，感触颇深。

在我看来，AI 的各种技术目前都还在起步阶段，相对成熟的推荐系统也仅仅是通过有限的用户浏览历史来进行简单的预测，而大数据时代的到来不仅意味着个人数据的广泛搜集和使用，更为计算机的算力提升和相应模型复杂度增加提供了契机。在未来，AI 必然会更进一步，做的更好，我也希望能够参与其中。

致谢

感谢《人工智能》课程 孔令和老师、许岩岩老师的精彩授课和悉心指导，段勇帅助教也为我提供了许多帮助。





上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

感谢聆听

饮水思源 爱国荣校