CSLT TECHNICAL REPORT- [Monday 28th November, 2016]

基于memory模型不同风格的诗词生成

Jiyuan Zhang[??,??] and Dong Wang[??,??*]

*Correspondence:
liuc@cslt.riit.tsinghua.edu.cn
??Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China
Full list of author information is available at the end of the article

Keywords: 诗词生成; 风格; memory模型; attention模型

1 背景介绍

诗词生成任务是让计算机自动创作诗词的技术。诗词生成一直被认为是一种高难度，依赖人类突发灵感的高级思维活动。另一方面，诗词创作又需要遵循严格的规律，如平仄，韵律等。这意味着生成诗词是一项既要循规蹈矩，又要寻求新意的艰苦劳动，仅有对平仄，韵律，意境等具有敏锐感觉，且思维活跃度极高的少数人能够胜任。幸运的是，这种这种在严格框架下进行有限创新的工作，计算机具有天然优势，它可以充分保证生成作品的合规性，同时在合规下探索各种可能的创新。让计算机自动生成诗词，可极大减少人类进行诗词创作的工作量，且有望产生挣脱传统思路束缚新颖诗词。即便用机器生成的诗词还不能与人类的诗人相比，但机器作品可以为人类提供候选或初级作品，使诗人创作更加容易；同时，计算机生成的诗词还可以为诗人提供灵感和刺激，激发他们不断创造新的音乐，仿止因长期创作带来的风格惰性和思维困顿，帮助诗人永褒创作青春。因此，自动诗词生成具有非常广阔的应用前景。

2 实验介绍

在本次实验中，主要的任务是对已有基于attention的诗词模型改进，使其能生成不同风格的诗词。一个诗人想写出不同风格的诗词，会在自己的记忆中寻找不同风格的诗，然后在平仄，韵律，主题的限定下，创作出不同风格的诗词。基于这个创作过程，我们想到用memory模型表示存储不同风格的诗，attention 模型表示学到的规则（平仄，韵律等），然后通过某种方式把这两个模型结合起来，进而产生不同风格的诗词。

3 实验原理及参数

本次实验使用两个模型，分别是attention和memory模型，下面就这两个模型做一个简单的介绍。

3.1 attention模型

attention机制就是在decoder过程中，可以动态的参考encoder不同部分。这个过程像人类在阅读时根据关注点的不同，聚焦在不同的段落或者句子。attention模型最先在神经机器翻译中应用，取得不错的效果。我们实把attention机制用在了诗词生成中，取得很好的效果，论文被ijcar收录。下图1展现attention机制：

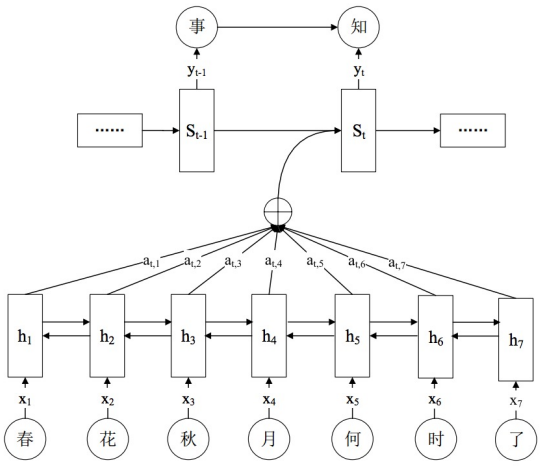

Figure 1: attention模型

3.2 memory模型

memory模型可以认为是对lstm的扩展，可以更加高效地筛选相关的信息，在QA 和LM中取得不错的效果。下面介绍一下memory模型中在QA的工作机制:

1 输入一个问题q及相应的上下文xi

2 上下文xi通过通过两个矩阵作为memory的输入和输出

3 问题q与memory的输入进行点积，通过sofemax计算出概率，与memory的输出点乘

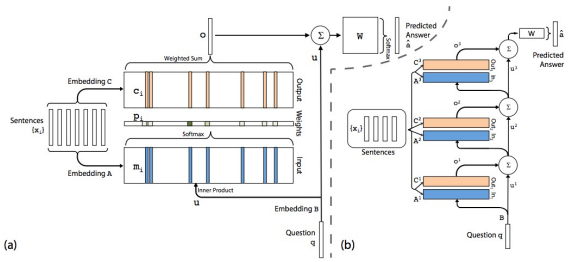

Figure 2: memory模型

4 实验输入及参数

- 数据源为58k首五言和七言的诗词，100首边塞诗和100首言情诗，其中100首边塞诗和100首言情诗分别通过运算形成两个不同风格的memory。
- encoder-decoder的参数为500
- attention的参数为1000
- maxout的参数为500

5 实验步骤

实验最终目的是探索一个attention和memory模型结合的方式，已达到生成不同风格的诗词的目的。

不过，我们现阶段的任务是证明memory模型有效，并探索两个模型结合方式。目前为止，我们总共设计了三种比较简单的改进方式，分别为encoder-memory，decoder-memory和在decoder-memory基础上结合方式的改变。

5.1 实验预备

这三种方式是在相同的基础上做出的改进。此基础是使用相同的attention模型，attention模型不进行更新。

5.2 改进一：encoder-memory

encoder-memory是attention模型中encoder的部分生成，下面是encoder-memory的基本过程：

1 与当前decoder的隐层通过attention的方式计算出memory中每个state对应的权重

2 让计算出的权重与其对应的memorywordvector相乘，得到权重化的wordvector

3 将其相加得到memory的最终输出m

4 m经过一个转换为与decoder隐层相同的维度，与其相加

encoder-memory结构，如下图3所示：

结论：并不能生成不同风格的诗词，因为最后得到的m对结果的影响很有限。

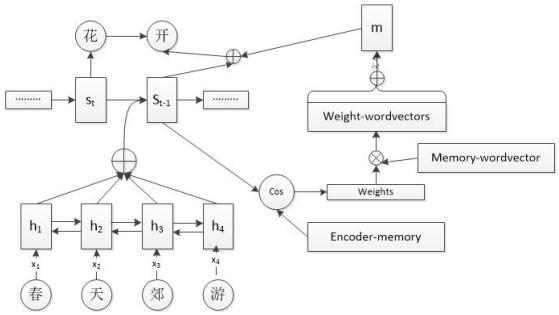

Figure 3: encoder-memory模型

5.3 改进二：decoder-memory

decoder-memory是attention模型中decoder的部分生成，下面是decoder-memory的基本过程：

1 与当前decoder的隐层通过cos的方式计算出memory中每个state对应的权重

2 让计算出的权重与其对应的memorywordvector相乘，得到权重化的wordvector

3 将其相加得到memory的最终输出m

4 m和decoder的隐层相加

decoder-memory结构，如下图4所示：

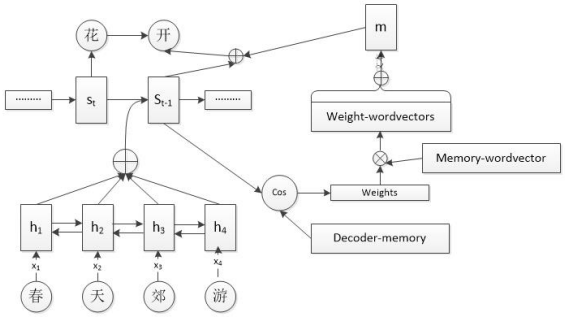

Figure 4: decoder-memory模型

结论：并不能生成不同风格的诗词，因为最后得到的m对结果的影响很有限。

5.4 改进三：

decoder-memory是attention模型中decoder的部分生成，下面是decoder-memory的基本过程：

1 与当前decoder的隐层通过cos的方式计算出memory中每个state对应的权重

2 让计算出的权重

3 选出权重最大的wordvector

4 用这个wordvector与整个词表做点积

decoder-memory结构，如下图5所示：

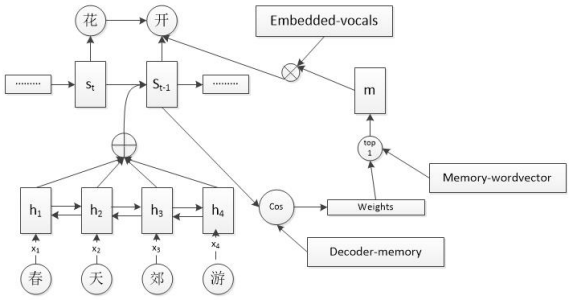

Figure 5: decoder-memory-diff-way模型

结论：能生成不同风格的诗词，以这种方式结合memory中的信息可以影响生成的结果。

示例：输入关键字“胡”，用边塞诗和言情诗的memory分别得到的结果：

边塞:一身不复无穷地 蓟口城边古木风 夜半胡笳声里雪 何须马上送君翁

言情:一身不复蝴蜂去 玉髻无人莫道啼 十二年来何处士 妆儿未免却相迷

Appendix A

这个附录将介绍一下诗词生成的代码的各个模块。

A.1 总目录

图6为总目录结构,下面为目录中的文件做出解释:

- seq2seq.py：负责attention模型和memory模型的创建和结合，是最核心的文件
- seq2seq_model.py：负责为模型的创建、生成与模型相匹配的数据格式和运行模型提供对外的接口
- train.py:负责训练模型的创建，主要功能是控制模型的参数
- predict.py：负责预测模型的创建，主要功能是控制模型的输出（在输出的过程加入规则的限制）
- run.py和rnn_cell.py：第一个文件提供了不同种类的rnn（比如，单项和双向rnn），第二个文件为rnn提供了不同的cell （比如，gru和lstm）
- GlobalParams.py:调用theano平台的各个参数初始值，作用为了保持与theano平台的代码效果完全一致
- memoryModule_decoder.py：用attention模型的decoder部分对不同风格的诗生成不同风格的memory
- memory.npy和memoryWordVector.npy：第一个文件是保存为numpy 数组形式的memory，第二个文件是memory中对应的wordvector
- 文件夹说明：memory_resource存放不同风格的诗，predict_resource存放与预测模型相关的文件；results存放预测模型的结果；tmp存放不同的模型参数；train_resource存放与训练相关的文件

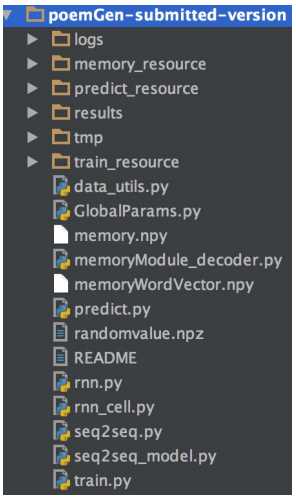

Figure 6: 总目录

A.2 seq2seq.py

seq2seq.py主要由model_with_buckets函数、sequence_loss函数、sequence_loss_by_example函数、embedding_attention_seq2seq 函数和attention_decoder函数组成。下面对这几个函数做一一介绍：

- model_with_buckets函数：为每个bucket建立图模型以及与之相关联的cost，这些图模型之间参数共享。tensorflow 的seq2seq 模型使用了bucket 机制（详情参看tensorflow 官方教程），该机制主要是为了减少在训练过程中冗余计算，进而加快训练速度。
- sequence_loss函数：计算batch中所有序列的平均cost
- sequence_loss_by_example函数：计算batch中每个序列的cost
- embedding_attention_seq2seq 函数：建立encoder模型
- attention_decoder函数:建立decoder模型、attention模型和memory模型

A.3 seq2seq_model.py

seq2seq_model.py由Seq2SeqModel类组成，而Seq2SeqModel类由__init__函数、step函数和get_batch函数组成。下面对这几个函数做一一介绍：

- __init__函数：创建图模型，并为模型配置参数
- step函数：每次处理的数据量为一个batch大小
- get_batch函数：加工数据为模型要求相匹配的格式

A.4 train.py

train.py由全局变量、tf.app.flags函数、read_data函数、create_model函数和train函数组成。下面对全局变量和这几个函数做一一介绍：

- 全局变量：包含基础公共的变量，如word2id，id2word，embeding_word
- tf.app.flags函数：创建公共变量，主要是为了创建模型的参数
- read_data函数：利用全局变量，处理训练数据为id形式
- create_model函数：创建seq2seqModel实例和模型并初始化
- train函数：按逻辑顺利调用上面函数，进行训练

A.5 predict.py

predict.py由全局变量、tf.app.flags函数、get_next_id函数、yunLv类、create_model函数、prepare_data_for_prediction函数、predict_process函数和predict函数组成。下面对全局变量和这几个函数做一一介绍：

- 全局变量：包含基础公共的变量，如word2id，id2word，embeding_word
- tf.app.flags函数：创建公共变量，主要是为了创建模型的参数
- prepare_data_for_prediction函数：利用全局变量，处理预测数据为id形式

- create_model函数：创建seq2seqModel实例和模型并初始化
- get_next_id函数：对每预测一个单词进行按限制条件（如，平仄、韵律等）筛选
- yunLv类：提供为每个单词查询平仄，以及两个单词是否押韵的功能
- predict_process：根据预测参数的设置，在预测中控制大局，如预测五言还是七言
- predict函数：按逻辑顺利调用上面函数，进行预测

A.6 memoryModule_decoder.py

train.py由全局变量、read_data函数、memNN函数和main函数组成。下面对全局变量和这几个函数做一一介绍：

- 全局变量：包含基础公共的变量，如word2id，id2word，embeding_word
- read_data函数：利用全局变量，处理训练数据为id形式
- memNN函数：创建模型
- main函数：初始化，运行模型并把结果保存下来作为memory