智能应答机器人——阶段总结

# 一、算法理论基础

关于智能应答理论基础的概述可以参照：

<http://blog.csdn.net/malefactor/article/details/51901115>

1、自动应答机器人主要分为两大类：

（1）检索式

检索式聊天机器人指的事先存在一个对话库，聊天系统接收到用户输入句子后，通过在对话库中**以搜索匹配的方式进行应答内容提取**，很明显这种方式对对话库要求很高，需要对话库足够大，能够尽量多地匹配用户问句，否则**会经常出现找不到合适回答**内容的情形，因为在真实场景下用户说什么都是可能的，但是**它的好处**是回答质量高，因为对话库中的内容都是真实的对话数据，表达比较自然。

（2）生成式

生成式聊天机器人则采取不同的技术思路，在接收到用户输入句子后，采用一定技术手段自动生成一句话作为应答，这个路线的机器人的**好处是**可能覆盖任意话题的用户问句，但是**缺点是**生成应答句子质量很可能会存在问题，比如可能存在语句不通顺存在句法错误等看上去比较低级的错误。

故：通过深度学习技术可以构建开放领域、生成式的聊天机器人。

2、几种主流技术思路

**2.1 基于人工模板的聊天机器人：**

通过人工设定对话场景，并对每个场景写一些针对性的对话模板，模板描述了用户可能的问题以及对应的答案模板。这个技术路线的好处是精准，缺点是需要大量人工工作，而且可扩展性差，需要一个场景一个场景去扩展。应该说目前市场上各种类似于Siri的对话机器人中都大量使用了人工模板的技术，主要是其精准性是其他方法还无法比拟的。

**2.2 基于检索技术的聊天机器人：（可以利用CNN匹配问答句，进而实现答句的检索）**

走的是类似搜索引擎的路线，事先存储好对话库并建立索引，根据用户问句，在对话库中进行模糊匹配找到最合适的应答内容。

**2.3  基于深度学习的聊天机器人：**

绝大多数技术都是在Encoder-Decoder（或者称作是Sequence to Sequence）深度学习技术框架下进行改进的。使用深度学习技术来开发聊天机器人相对传统方法来说整体思路是非常简单可扩展的。

3、深度学习主要算法

**3.1 卷积神经网络（CNN）——监督式学习**

CNN的意义：加权叠加

把输入信号各个时间点的单位相应加权叠加，直到得到输出信号，可以用来处理语音信号、图像、文本，**如今应用最多的是应用CNN对图片进行识别**，具体代码可以参照project包里面的程序，其中**C:\Users\IA-weixin200\PycharmProjects**\project\test.py则是对于自己手写的图片进行识别，该程序里面已经写好程序对手写图片格式进行二值图转化，存储在28pix文件夹中，具体代码均有详细的注释。mnist\_train.py是训练模型，对训练集进行训练，mnist\_eval.py是测试模型，对测试数据进行测试，test.py则是一个独立的测试+训练模型，代码均有详细备注。

在现有的研究中CNN多用于图像识别，也可用于文本分析，在处理NPL时，输入不再是像素点，而是矩阵表示的文档。

CNN应用于NLP，多用于分类任务、语义分析、话题分类，值得注意的有以下几点：

1. 在做卷积时，对于矩阵边缘的元素，可采用补零法，即落在矩阵范围外的均设为0，如此称为宽卷积；不使用补零的为窄卷积；

宽卷积长度=（输入+2\*间隔—滤波器长）+1 （具体参照上面文章）

1. 对于步长：选择更大的步长会让模型更加接近于RNN
2. 池化：在NLP中，通常对整个输出做池化，最终得到一个输出
3. 通道：把不同类的词向量表征看做独立的通道
4. 输入表征选择：word2vec，one-hot，Glovec……

CNN卷积和池化会丢失局部位置信息，所以在处理自然语言时具有一定的局现性。

**可以调节的参数：**

输入表征、滤波器大小及数量、池化策略（取最大值优于取平均值）、（ReLU、tanh、sigmoid、softplus……）

**CNN参考资料：**

//利用CNN进行语义分析

<http://www.cnblogs.com/anyview/p/5106655.html>

<http://blog.csdn.net/malefactor/article/details/50374237>

//利用TensorFlow实现卷积

<http://news.ifeng.com/a/20170515/51096526_0.shtml>

<http://blog.csdn.net/NNNNNNNNNNNNY/article/details/70216265>

<http://www.cnblogs.com/weizhen/p/6741702.html>

**3.2 循环神经网络（RNN）**

RNN从定义上讲分为两类：recurent（时间循环）和recursive（结构循环），目前所说的均是Recurent，Recursive结构复杂，不易应用，鲜少有应用的。

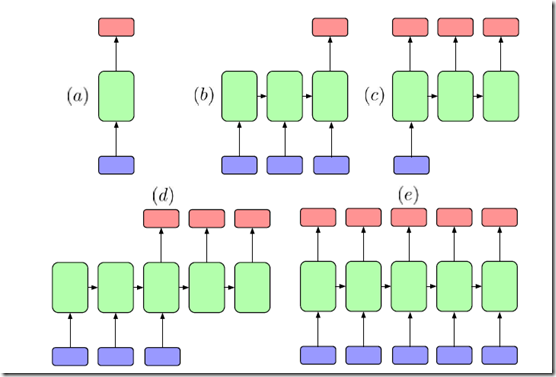
时间循环神经网络采用**链式结构**，按照句子中词的顺序进行排序，句子后面的词语前面的词有关。

1. **)**

是空间维度的展开，是一个**树结构**，用recurxive neural network来建模的话，就是假设句子是一个树状结构，由几个部分(主语，谓语，宾语）组成，而每个部分又可以在分成几个小部分，即某一部分的信息由它的子树的信息组合而来，整句话的信息由组成这句话的几个部分组合而来。

参考资料：<https://www.zhihu.com/question/36824148>

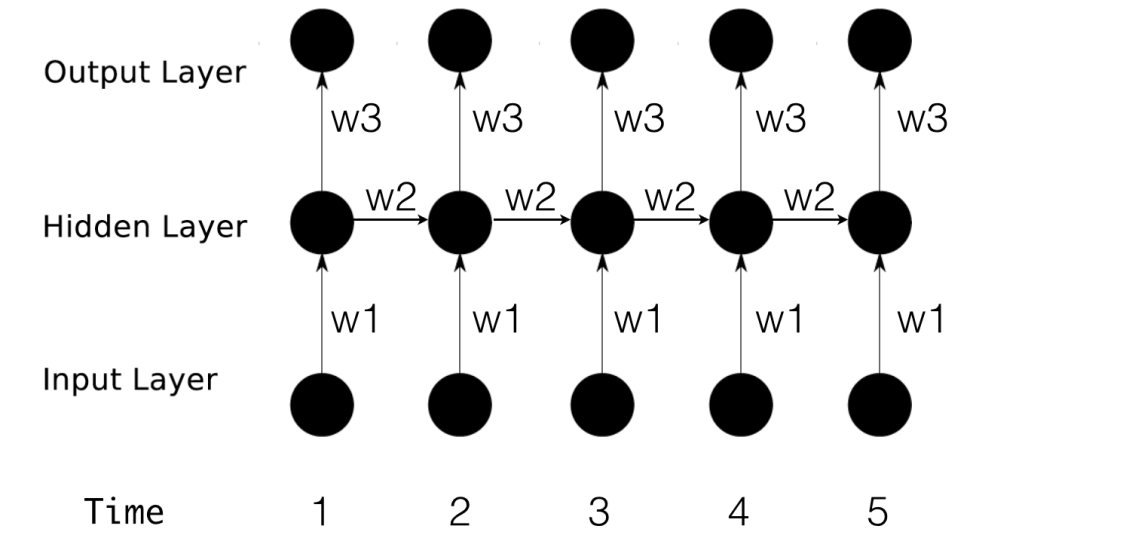
下面详细介绍RNN（时间循环神经网络）—**具体原理参照笔记前几页**



**对于RNN输出问题：**

RNN的样本可以是如下形式的：1）输入输出均为序列；2）输入为序列，输出为样本标签；3）输入单个样本，输出为序列。

图中蓝色节点为输入，红色为输出，绿色为 hidden node 或者 memory block ，（a）描述的是传统的 MLP 网络，即不考虑数据的时序性，认为数据是独立的；**（b）将输入序列映射为一个定长向量（分类标签），可用于文本、视频分类**；（c）输入为单个数据点，输出为序列数据，典型的代表为 Image Captioning；**（d）这种是一种结构 sequence-to-sequence 的任务，常被用于机器翻译，两个序列长度不一定相等；**（e）这种结构会得到一个文本的生成模型，每词都会预测下一时刻的字符。



RNN本质是输出概率最大的一个词，所以RNN根据输出的形式不同，其输出层、损失函数设定就不同：（1）对于字母级别的，每个时刻都要输出，预测下一时刻应该输出的字母；（2）对于句子：只需要知道最后时刻的输出即可，也即预测句子后出现概率最大的一个词是什么，之前每个时刻的输出均表示一个状态，最后一个时刻判断完整语句，进行预测；（3）对于问答：则需要应用seq2seq，应用两个RNN的原因：（个人理解）对于应答，需要输入完整句子之后，根据之前时刻的输出预测下一时刻的输出，再将每一个时刻的输出进行组合，得到应答句子。

**RNN讲解资料参考：**

<https://baijia.baidu.com/s?old_id=584210>

<http://www.cnblogs.com/ooon/p/5603869.html>

<http://blog.csdn.net/u014595019/article/details/52826423>

<http://www.iteye.com/news/31672>

<http://blog.csdn.net/lyc_yongcai/article/details/73201366>

<http://www.sohu.com/a/133495402_610300>

<http://blog.csdn.net/lyc_yongcai/article/details/73201366>

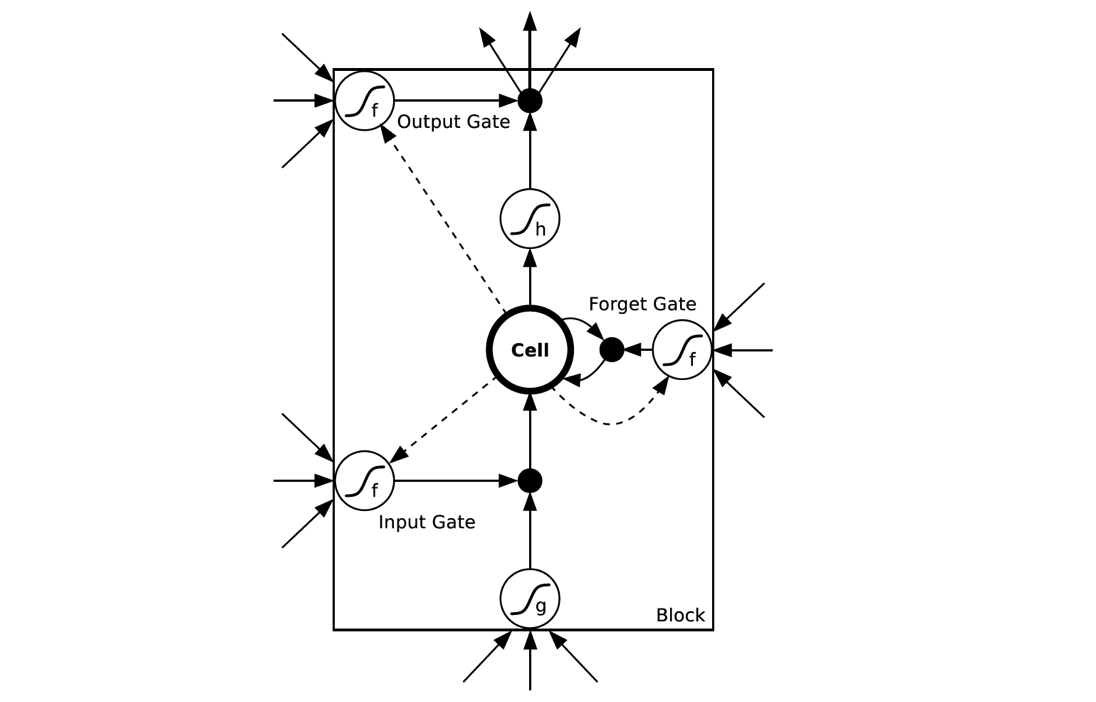
<http://blog.csdn.net/fangqingan_java/article/details/53232415>

//利用rnn实现语义分析

<http://blog.csdn.net/malefactor/article/details/50878936>

<http://www.ctiforum.com/news/guandian/510101.html>

**3.3 LSTM长短期记忆**



原生的RNN会遇到一个很大的问题：后面时间的节点对于前面时间的节点感知力下降，也就是忘事儿。LSTM是RNN一种变换，用上述的记忆单元替代神经元。其中：

cell：表示state （具体公式解析见笔记）

输入门：控制新加入多少信息

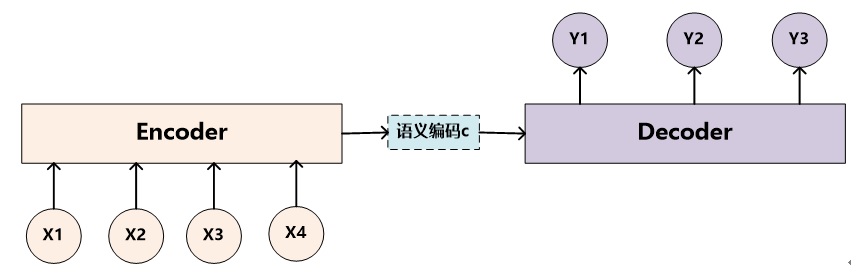
遗忘门：控制一个记忆单元遗忘多少信息

输出门：表示输出多少信息

4、自动应答框架

**4.1 Encoder—Decoder**

目前对于开放领域生成式聊天机器人技术而言，多数技术采用了Encoder-Decoder框架。



Encoder-Decoder框架可以看作是一种文本处理领域的研究模式，应用场景异常广泛，不仅仅可以用在对话机器人领域，还可以应用在机器翻译、文本摘要、句法分析等各种场合。上图是文本处理领域里常用的Encoder-Decoder框架最抽象的一种表示。具体实现的时候，编码器和解码器都不是固定的,可选的有CNN/RNN/BiRNN/GRU/LSTM等等，你可以自由组合。比如说，你在编码时使用BiRNN,解码时使用RNN，或者在编码时使用RNN,解码时使用LSTM。

对于聊天机器人来说，X指的是用户输入语句，一般称作Message，而Y一般指的是聊天机器人的应答语句，一般称作Response。其含义是当用户输入Message后，经过Encoder-Decoder框架计算，首先由Encoder对Message进行语义编码，形成中间语义表示C，Decoder根据中间语义表示C生成了聊天机器人的应答Response。这样，用户反复输入不同的Message，聊天机器人每次都形成新的应答Response，形成了一个实际的对话系统。

拿RNN作为编码、解码器进行讲解：

（1）当前时间的隐藏状态是由上一时间的状态和当前时间输入决定的



（2）获得了各个时间段的隐藏层以后，再将隐藏层的信息汇总，生成**最后的语义向量**



（3）解码阶段可以看做编码的逆过程。这个阶段，我们要根据给定的语义向量C和之前已经生成的输出序列y1,y2,…yt−1



故： 从上述可以看出，一个独立的RNN有一个输出层，将隐藏层的信息处理成输出；在Encoder—Decoder模型中，两个RNN直接是隐藏层连接，衔接部分的输出层去掉了，故Encoder—Decoder模型中只有一个输入和一个输出，中间的隐层连接成一个中间语义，这样训练的时候，问句放在Encoder，答句放在Decoder；此外，解码过程重复利用中间编码C，也即整个模型通过Encoder将整个句子输入，得到中间编码，解码时根据中间编码C和之前的输出得到此时刻的输出yt。故个人理解，之所以用两个模型组合使用，是为了获得中间编码，解码过程知道了整个输入的语义之后，根据历史输出得到下一个时刻输出。

**4.2 seq2seq模型（解决“安全回答”问题）**

在训练数据的时候经常会有很多“呵呵”、“好的”等无效对话，利用sequence – sequence可以解决这个问题，seq2eq模型在计算概率的时候加了一个优化目标函数，为了最大化互信息（MMI）：



MMI的优化目标除了最大化从Message生成应答Response的概率，同时加入了反向优化目标，即最大化应答Response产生Message的概率，其中lamda是控制两者哪个更重要的调节超参数。这个优化目标函数要求应答Response和Message内容密切相关而不仅仅是考虑哪个Response更高概率出现，所以降低了那些非常常见的回答的生成概率，使得应答Response更多样化且跟Message语义更相关。

谷歌新开源的seq2seq库：<https://github.com/google/seq2seq>

**4.3 框架优化—Attention模型**

encoder-decoder缺点：编码器要将整个序列的信息压缩进一个固定长度的向量中去。但是这样做有两个弊端，一是语义向量无法完全表示整个序列的信息，还有就是先输入的内容携带的信息会被后输入的信息稀释掉，或者说，被覆盖了。输入序列越长，这个现象就越严重。这就使得在解码的时候一开始就没有获得输入序列足够的信息。

如果输入句子比较长，此时所有语义完全通过一个中间语义向量来表示，单词自身的信息已经消失，可想而知会丢失很多细节信息，这也是为何要引入注意力模型的重要原因。AM模型成句当前生成单词，给出每个词对应句子的注意力分配概率信息。简单说，AM模型可以抽取重要主题。

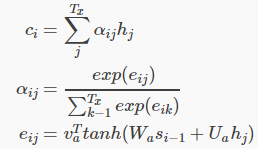
编码过程：（双向RNN）



解码过程:



其中：



参考资料：

关于Encoder-Decoder框架及Attention模型

http://blog.csdn.net/malefactor/article/details/50550211

http://weibo.com/fly51fly?noscale\_head=1&is\_search=1&key\_word=seq2seq&is\_all=1#\_0

http://www.360doc.com/content/16/0524/20/31115656\_561988998.shtml

http://www.2cto.com/kf/201612/575911.html

http://wiki.jikexueyuan.com/project/tensorflow-zh/tutorials/seq2seq.html

https://github.com/undersail/easybot

http://blog.csdn.net/sunlylorn/article/details/50607376

http://blog.csdn.net/u014595019/article/details/52826423

http://blog.csdn.net/fangqingan\_java/article/details/53232415

关于TensorFlow seq2seq实现聊天机器人

http://www.easyapple.net/?p=1384

http://www.360doc.com/content/17/0413/01/41269240\_645179772.shtml

http://www.sohu.com/a/128795695\_465975

https://github.com/henriblancke/neural-chat

http://blog.csdn.net/mebiuw/article/details/52749822

https://github.com/google/seq2seq/tree/master/seq2seq

TensorFlow实现RNN

<http://www.cnblogs.com/hellocwh/p/5626457.html>

自动应答机器人

<http://www.shareditor.com/blogshow/?blogId=104>

5、理论中的易混知识点

**5.1 卷积中的batch**

Batch是分批处理，为了能够更好地处理非凸损失函数，这样能够合理的使用内存。在非凸情况下，卷积结果容易卡在局部最优，批处理是对全样本部分抽样。

Batch：决定了下降的方向，如果数据集小，可以采用全数聚集；当数据集过大，内存受到限制，一次性载入所有数据不可行，各个batch间差异大，梯度修正值会发生抵消，无法发挥修正作用，故利用批梯度下降法来训练参数。

合理范围内增大batch\_size，有以下好处：

1. 内存利用率提高，并行化效率提高；
2. 跑完一次epoch（全数据集）迭代次数减少，处理速度提高；
3. Batch\_size越大，下降方向越准，引起震荡越小；

Batch\_size达到某值时，达到收敛精度的最优，当batch\_size过大时，会出现以下问题：

1. 内存容量不够大；
2. 参数修正缓慢，达到相同精度所花时间增加；
3. Batch达到某值时，下降方向不再发生变化

对batch进行调参时，看batch会不会使loss函数变小。

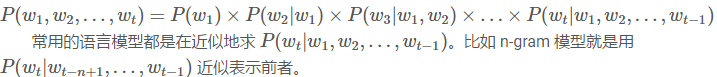
**5.2 seq2seq模型源代码中的bucketing策略**

在seq2seq中，有一个bucketing策略，在正常情况下，计算机输入输出长度固定，但是句子长度不固定，所以这样可能会发生句子长度短，浪费内存，或者句子长度长，输入输出不够的情况，故在seq2seq中设置若干个buckets，每个bucket指定一个输入和输出长度。如：buckets = [(5, 10), (10, 15), (20, 25), (40, 50)]，将训练样本分成4份，训练参数时，从4个中随机选一个，并选batch个训练样例进行训练。（有关seq2seq模型，源代码赋有解析）

关于正则化的讲解：<http://blog.csdn.net/ztf312/article/details/50894115>

# 二、语言模型理论基础

1、语言模型



表示t个词按照顺序构成一句话的概率，详细的不在介绍，具体可以参考：

<http://licstar.net/archives/328#s0>

词向量：独热编码（one-hot编码）会出现“词汇鸿沟”，任意两个词是孤立的，无法查找词之间的相关性，以及反义词、同义词等。词向量让相关或者相似的词，在距离上更接近。

利用word2vec从语料得到词向量主要步骤：

1. 分词：可利用jieba分词
2. 统计词频：去掉高频词（无特殊性）、去掉低频词（无普适性）
3. 构建哈弗曼树：对整个词典构建哈弗曼树，叶子节点为词典中的词
4. 训练：语料小的时候可以用skip-gram模型，语料大的时候用CBOW模型

关于word2vec的代码解析可以参考：

<http://blog.csdn.net/lingerlanlan/article/details/38232755>

<http://blog.csdn.net/messiandzcy/article/details/44100391>

<http://blog.csdn.net/fangpinlei/article/details/52200832>

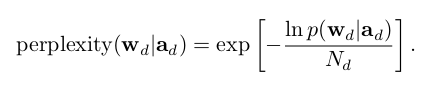
2、word2vec理论

略

参考资料：<https://www.zhihu.com/question/29894719>

3、语言模型评价标准

* 实用方法：通过查看该模型在实际应用（如拼写检查、机器翻译）中的表现来评价，优点是直观、实用，缺点是缺乏针对性、不够客观；
* 理论方法：迷惑度/困惑度/混乱度（preplexity），其基本思想是给测试集的句子赋予较高概率值的语言模型较好,当语言模型训练完之后，测试集中的句子都是正常的句子，那么训练好的模型就是在测试集上的概率越高越好



概率越大，语言模型也好，由公式可知，困惑度越小，句子概率越大，语言模型越好。

具体参考资料：

<http://blog.csdn.net/lujiandong1/article/details/53259653>

<http://blog.sina.com.cn/s/blog_13647fd0f0102w7t7.html>

<http://blog.csdn.net/xiaokang06/article/details/17965965>

# 三、实际操作

1、TensorFlow安装问题

**1.1 win7 64位下安装TensorFlow**

**Win7下安装的是Python3.6+TensorFlow1.0**

参考资料：

**//tensorflow安装**

<http://blog.csdn.net/u013080652/article/details/68922702>

**TensorFlow安装cpu版本遇见困难，解决办法:加入以下语句即可**

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL']='2'

**（Python学习---知乎指路）知乎上提供了很多可以供你学习的网站、书籍**

<https://www.zhihu.com/question/20702054>

**//TensorFlow官方教程**

<http://www.tensorfly.cn/tfdoc/get_started/basic_usage.html>

**1.2 Linux 64位下安装TensorFlow（Linux下安装的是Python2.7+TensorFlow0.8）**

参考资料：

**在linux下安装TensorFlow遇见的问题及解决办法**

<http://www.cnblogs.com/chenjack/p/6322680.html>

<http://blog.csdn.net/levy_cui/article/details/51251095>

2、初步代码实现

**2.1 代码位置**

所有程序存放在：C:\Users\IA-weixin200\PycharmProjects

其中：

1. project 存放的是CNN手写识别的代码，test是一个独立的程序，训练完数据之后，识别自己制作的手写图片；其他三个程序则是一套，将训练、测试、反向传播算法分开。
2. AI-No.1 存放的是现在已经运行出来的应答雏形，需要在Linux下运行，支持TensorFlow0.8+Python2.7，代码已经调好，如果使用word2vec产生词向量，则无需运行fenci.py，直接将train.py的词向量路径改了应该就可以。

可参考：<http://www.cnblogs.com/LittleHann/p/6426610.html>

<http://www.voidcn.com/blog/u014365862/article/p-6355768.html>

1. easybot-master 存放的是第一次运行的程序，因为没有运行到底，所以不知道最终结果如何，如果需要改进成app的话，可以使用这个例子，参考：<http://www.easyapple.net/?p=1384>

参考资料：

<http://www.shareditor.com/blogshow/?blogId=121&hmsr=toutiao.io&utm_medium=toutiao.io&utm_source=toutiao.io>

<http://www.sohu.com/a/147285889_609569>

**2.2 Linux下操作流程**

由于在安装Python的时候，没将Python2.7安装到默认路径，所以在连上linux服务器之后，**首先设置路径**：

ls -1 $HOME/py27/bin/python\*

export PATH=$HOME/py27/bin:$PATH

which python

**之后就可以找到文件，运行程序**

cd py27/bin/AI-No.1

python train.py

训练完成之后，可以进行测试

python test.py

3、模型优化及构想

**3.1 jieba分词**

参考资料：<http://blog.csdn.net/john_xyz/article/details/54645527>

**3.2 Word2vec生成词向量**

关于word2vec的一些实现：

<http://www.cnblogs.com/hebin/p/3507609.html>

**3.3 训练模型并调参**

训练模型时很多参数可以调整，例如在word2vec中可以选择不同的模型，CBOW对大语料好，Skip-gram则在语料较小时使用有优势，在模型中加入词序，对模型提升不大；

对于语料，有研究说语料对词向量的影响比模型对词向量的影响大，语料越大越好；

参数方面，word2vec中迭代次数、词向量长度、词向量维度都可以调整

在seq2seq原始模型中，很多参数也可以调整，如：

layer\_size = 256 # 每层大小

num\_layers = 3 # 层数

batch\_size = 64

对于学习率，现有AI-No.1模型采用指数衰减的方法调整学习率，可以自己设置学习率衰减因子，控制衰减速度

根据语料库调整参数，以此来优化模型。

一些关于深度学习的学习资料：E:\深度学习

# 四、总结

路漫漫其修远兮 吾将上下而求索。对于一个完善的智能问答系统还有很多需要做，分词、语料的训练、参数的调整，理论部分也还有很多不成熟的地方，但是，学习过程是快乐的，加油！