《程序设计基础》
AI 诗人实验报告

目录

一、作业概述

<1>项目目的

主要通过基于词频分析和基于 tensorflow 的深度学习两种方法让程序自动写诗，并通过写诗的结果来分析写诗的原理以及两种写诗方法的异同。

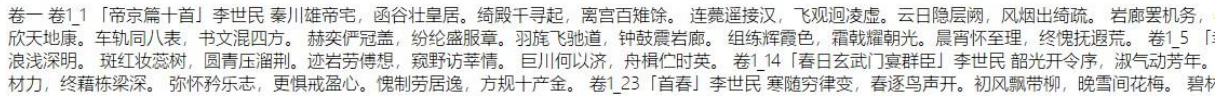
<2>项目内容

1.基于词频分析

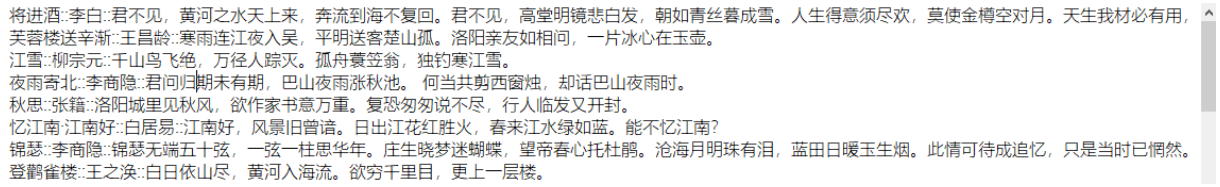
（1）爬虫

主要文件：__init__.py

从 http://www.gushiwen.org 网站爬取诗句，用于词频分析时格式如下：


卷一 卷1_1「帝京篇十首」李世民 秦川雄帝宅，函谷壮皇居。绮殿千寻起，离宫百雉馀。 连薨遥接汉，飞观迥凌虚。云日隐层阙，风烟出绮疏。 岩廊罢机务，
欣天地康。车轨同八表，书文混四方。 赫奕俨冠盖，纷纶盛服章。羽旄飞驰道，钟鼓震岩廊。 组练辉霞色，霜戟耀朝光。晨胥怀至理，终愧抚遐荒。 卷1_5「
浪浅深明。 斑红妆蕊树，圆青压溜荆。迹岩劳傅想，窥野访莘情。 巨川何以济，舟楫仁时英。 卷1_14「春日玄武门宴群臣」李世民 韶光开令序，淑气动芳年。
材力，终藉栋梁深。 弥怀矜乐志，更惧戒盈心。愧制劳居逸，方规十产金。 卷1_23「首春」李世民 寒随穷律变，春逐鸟声开。初风飘带柳，晚雪间花梅。 碧林

用于 tensorflow 作诗时格式如下：


将进酒::李白::君不见，黄河之水天上来，奔流到海不复回。君不见，高堂明镜悲白发，朝如青丝暮成雪。人生得意须尽欢，莫使金樽空对月。天生我材必有用，
芙蓉楼送辛渐::王昌龄::寒雨连江夜入吴，平明送客楚山孤。洛阳亲友如相问，一片冰心在玉壶。
江雪::柳宗元::千山鸟飞绝，万径人踪灭。孤舟蓑笠翁，独钓寒江雪。
夜雨寄北::李商隐::君问归期未有期，巴山夜雨涨秋池。 何当共剪西窗烛，却话巴山夜雨时。
秋思::张籍::洛阳城里见秋风，欲作家书意万重。复恐匆匆说不尽，行人临发又开封。
忆江南·江南好::白居易::江南好，风景旧曾谙。日出江花红胜火，春来江水绿如蓝。能不忆江南?
锦瑟::李商隐::锦瑟无端五十弦，一弦一柱思华年。庄生晓梦迷蝴蝶，望帝春心托杜鹃。沧海月明珠有泪，蓝田日暖玉生烟。此情可待成追忆，只是当时已惘然。
登鹳雀楼::王之涣::白日依山尽，黄河入海流。欲穷千里目，更上一层楼。

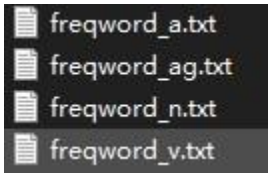
代码如下：

```python
import urllib3
from bs4 import BeautifulSoup
import certifi
import time
import random
import re

rex=re.compile(r'\(.*?\)')

f = open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/s_song.txt', "w+",encoding="utf-8")
http = urllib3.PoolManager(
    cert_reqs='CERT_REQUIRED',
    ca_certs=certifi.where())

for poemId in range(1, 1001):
    print(poemId)
    url ='https://www.gushiwen.org/shiwen/default_3A888888888888A'+str(poemId)+'.aspx'
    #'http://www.gushiwen.org/wen_' + str(poemId) + '.aspx' 原始
    #content = soup.find('div', class_="contson").get_text()
    #file.write(content)

    #'https://www.gushiwen.org/shiwen/default.aspx?page=' + str(poemId) 所有诗
    #'https://www.gushiwen.org/shiwen/default_3A6666666666666A'+str(poemId)+'.aspx'   唐诗
    #'https://www.gushiwen.org/shiwen/default_3A888888888888A'+str(poemId)+'.aspx'   宋诗
    r = http.request('GET', url)
    soup = BeautifulSoup(r.data, 'html.parser')
    content = soup.find('div', class_="contson").get_text()
    content = re.sub(rex, "", content)
    content = content.replace('\n', '')
    author = soup.find('p', class_="source").get_text()
    author.replace('\n', '')
    author=author[author.find(": ")+1:]
    title = soup.find('b').get_text()
    title.replace('\n', '')
    result=(title+'::'+author+'::'+content+'\n')
    # if poemId >= 1109 and poemId < 2010:
    #     file.write(content)
    f.write(result)
    time.sleep(random.random()*3)
f.close()
```

与老师所给代码不同的是，此处我使用了 urllib3 库而不是 requests 库

（2）高频词统计

主要文件：dataHandler.py
使用 jieba 库，统计形容词，副词，名称以及动词并且提取其高频词



（3）数据处理

主要文件：zzcf.py mlzzcf.py
Zzcf.py 使用正则表达式，在爬取的数据中仅提取诗句与标点，处理后格式如下：

秦川雄帝宅，函谷壮皇居。绮殿千寻起，离宫百雉馀。连薨遥接汉，飞观迥凌虚。云日隐层阙，风烟出绮疏。岩廊罢机务，崇文聊驻辇。玉匣启龙图，金绳披凤定，怀柔万国夷。梯山咸入款，驾海亦来思。单于陪武帐，端扆朝四岳，无为任百司。霜节明秋景，轻冰结水湄。芸黄遍原隰，禾颖积京畿。共乐还乡宴，代马烧火红。雕戈夏服箭，羽骑绿沉弓。怖兽潜幽壑，惊禽散翠空。长烟晦落景，灌木振严风。所为除民瘼，非是悦林丛。烈烈寒风起，惨惨飞云浮。霜浓凝广隰，已获千箱庆，何以继熏风。碧昏朝合雾，丹卷暝韬霞。结叶繁云色，凝琼遍雪华。光楼皎若粉，映幕集疑沙。泛柳飞飞絮，妆梅片片花。照壁台圆月，飘珠箔穿

Mlzzcf.py（原创）则是将数据处理为适用于基于 tensorflow 的作诗代码，可以添加进其源数据之中，处理后格式如下：

帝京篇十首::李世民::塞外悲风切，交河冰已结。瀚海百重波，阴山千里雪。迥戍危烽火，层峦引高节。悠悠寒沙连骑迹，朔吹断边声。胡尘清玉塞，羌笛韵金钲

饮马长城窟行::李世民::执契静三边，持衡临万姓。玉彩辉关烛，金华流日镜。无为宇宙清，有美璇玑正。皎佩星连景，飘衣云结庆。戢武耀七德，升文辉九功。

执契静三边::李世民::条风开献节，灰律动初阳。百蛮奉遐赆，万国朝未央。虽无舜禹迹，幸欣天地康。车轨同八表，书文混四方。赫奕俨冠盖，纷纶盛服章。羽

（4）生成五言律诗（主要调研对象）

主要文件：TangshiGene.py
代码如下：

```python
import random
import time
import pinyin


def search(array, target):
    # array为二维数组
    for index in range(len(array)):
        if target in array[index]:
            return True
    return False


# 生成五言律诗
def Line5():
    noun = open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/freqword_n.txt',
                encoding='utf-8').readlines()
    verb = open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/freqword_v.txt',
                encoding='utf-8').readlines()

    nounlist = []
    for word in noun:
        i = 0
        while i < len(word):
            if word[i:i + 2] != "\n" and word[i] != ' ' and word[i + 1] != ' ':
                nounlist.append(word[i:i + 2])
            i = i + 3

    verblist = []
    for word in verb:
        i = 0
        while i < len(word):
            if word[i:i + 2] != "\n" and word[i] != ' ' and word[i + 1] != ' ':
                verblist.append(word[i:i + 2])
            i = i + 3

    num = 0

    rhythm = ""
    rhythmList = [['a', 'ua', 'ia'], ['e', 'o', 'uo'], ['e', 'ie', 've'], ['i', 'v', 'er'], ['u'], ['ai', 'uai'], ['ei', 'ui'], ['ao', 'iao'],
```

```
问题  输出  调试控制台  终端

FileNotFoundError: [Errno 2] No such file or directory: 'D:/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/freqword_n.txt'
PS C:\Users\qq185\OneDrive - bupt.edu.cn\python\AI诗人-2018211604-2018213344-凌国瀚\基于词频作诗> python -u "c:\Users\qq185\OneDrive - bupt.edu.cn\python\AI诗人-20
"
镜花水月
清风来天地
花枝顾万国
水声道瑶台
月光应零落
PS C:\Users\qq185\OneDrive - bupt.edu.cn\python\AI诗人-2018211604-2018213344-凌国瀚\基于词频作诗>
```

（5）生成藏头诗

主要文件：TangshiGene2.py
此处仅需要输入藏头诗的第一个字组成的词语便可以生成
代码如下：

```python
import random
import pinyin

#生成藏头诗
def Line5_Head(str):
    noun = open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/freqword_n.txt', encoding='utf-8').readlines()
    verb = open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/freqword_v.txt', encoding='utf-8').readlines()

    nounlist = []
    for word in noun:
        i = 0
        while i<len(word):
            if word[i:i+2]!="\n":
                nounlist.append(word[i:i+2])
            i=i+3

    verblist = []
    for word in verb:
        i = 0
        while i<len(word):
            if word[i:i+2]!="\n":
                verblist.append(word[i:i+2])
            i=i+3

    # sentence = ""
    num = 0

    rhythm = ""
    rhythmList = ["a", "e", "i", "o", "u"]
    while num < 4:
        for x in range(len(nounlist)):
            if nounlist[x][0] == str[num]:
                i = x

        i1 = random.randint(1, len(nounlist)-1)
        j = random.randint(1, len(verblist)-1)

        ind = 0
        ind1 = 0
        if (num == 1):
```

（6）生成数据库

主要文件：toDB.py，shi.db
将使用的诗句（mlzzcf.txt）转换为数据库，便于处理
代码如下：

```python
import sqlite3

conn=sqlite3.connect('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/shi.db')
cur=conn.cursor()

file = open("C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/mlzzcf.txt", "r",encoding='UTF-8')
for line in file:  #every line is a poem
    title, author, poem = line.strip().split("::")  #get title and poem
    poem = poem.replace(' ','')
    if len(poem) < 10 or len(poem) > 512:  #filter poem
        continue
    if '_' in poem or '《' in poem or '[' in poem or '(' in poem or ' (' in poem:
        continue
    sql="insert into shi values(?,?,?)"
    data=(title,author,poem)
    cur.execute(sql,data)
    conn.commit()
file.close()
conn.close()
```

（7）生成词云

主要文件：wordcould.py, Figure_1.png
根据高频词生成词云
代码如下：

```python
import jieba.analyse
from os import path
from scipy.misc import imread
import matplotlib as mpl
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

if __name__ == "__main__":

    mpl.rcParams['font.sans-serif'] = ['FangSong']
    #mpl.rcParams['axes.unicode_minus'] = False

    content = open("C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/zzcf.txt","rb").read()

    # tags extraction based on TF-IDF algorithm
    tags = jieba.analyse.extract_tags(content, topK=100, withWeight=False)
    text =" ".join(tags)
    # read the mask
    d = path.dirname(__file__)
    trump_coloring = imread(path.join(d, "data\\Trump.jpg"))

    wc = WordCloud(font_path='C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/simsun.ttc',
            background_color="white", max_words=50, mask=trump_coloring,
            max_font_size=40, random_state=42)

    # generate word cloud
    wc.generate(text)

    # generate color from image
    image_colors = ImageColorGenerator(trump_coloring)

    plt.imshow(wc)
    plt.axis("off")
    plt.show()
```

效果见数据分析

（8）数据押韵分析

主要文件：rhymerate.py
统计源数据中押韵的诗句比例

```python
import sqlite3
import re
import pinyin

conn = sqlite3.connect(
    'C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/shi.db')
c = conn.cursor()
print("Opened database successfully")

cursor = c.execute("SELECT title,author,poem  from shi")
rhythmList = ["a", "e", "i", "o", "u"]
sum = 0
isrhtme = 0
for row in cursor:
    rhythm = []
    p1 = r"[\u4e00-\u9fa5]{5,7}[\u3002|\uff0c]"  # [汉字]{重复5-7次}[中文句号|中文逗号]
    pattern1 = re.compile(p1)
    result1 = pattern1.findall(row[2])
    if result1 == []:
        continue
    for x in result1:
        a = pinyin.get(x[-2], format="strip")
        for y in range(len(a) - 1, -1, -1):
            if a[y] in rhythmList:
                rhythm.append(a[y:])
    temp = list(set(rhythm))
    if len(temp) != len(rhythm):
        isrhtme += 1
    sum += 1
rate = (isrhtme / sum) * 100
print("The rhyme rate is "+"%.2f"%(rate)+"%")
print("Operation done successfully")
conn.close()
```

2. 基于 tensorflow 的机器学习

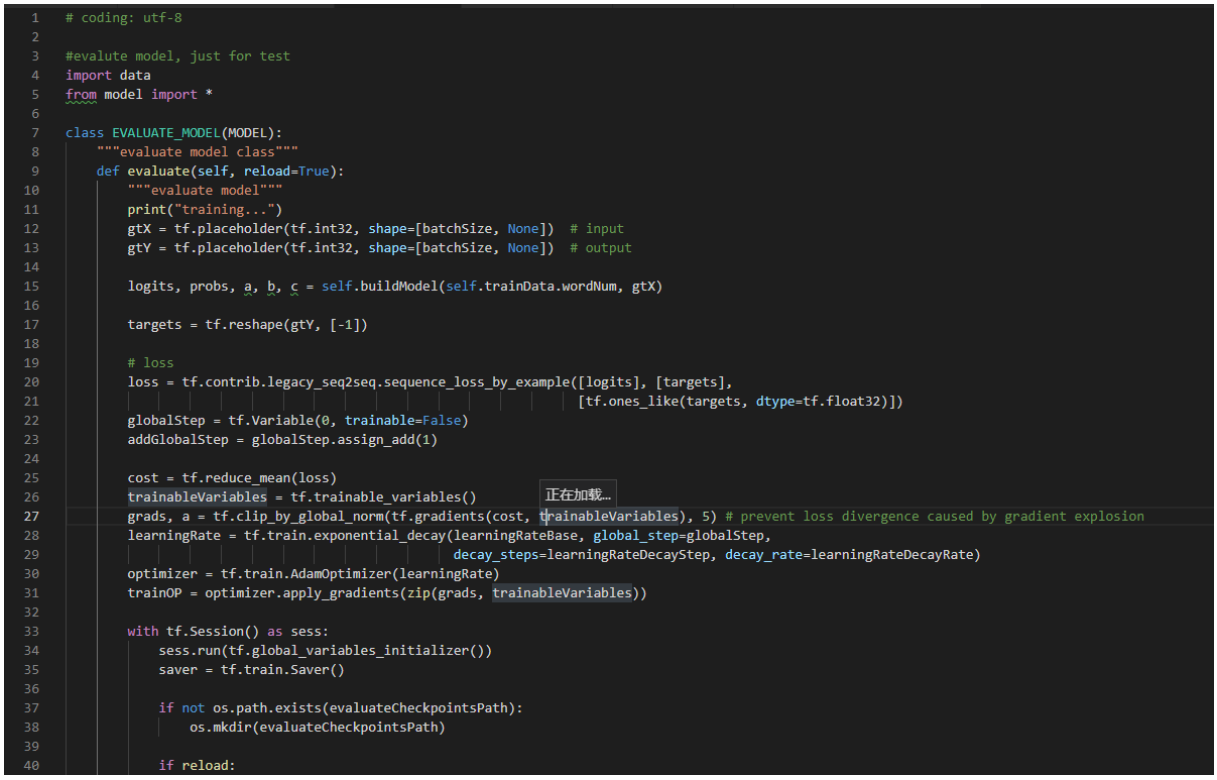
使用 python 的 tensorflow 库创建训练(train)、作诗(test)和藏头诗(head)的模型。在一个统一的项目 config.py 中设置相关参数（训练次数、训练诗句类型等）并调用相关库：

```python
# coding: utf-8

import tensorflow as tf
import numpy as np
import argparse
import os
import random
import time
import collections

batchSize = 64

learningRateBase = 0.001
learningRateDecayStep = 1000
learningRateDecayRate = 0.95

epochNum = 10                    # train epoch default=10
generateNum = 5                   # number of generated poems per time

type = "wangfeng"                   # dataset to use, poetrySong, shijing, songci, wangfeng, spider_shi etc
trainPoems = "./dataset/" + type + "/" + type + ".txt" # training file location
checkpointsPath = "./checkpoints/" + type # checkpoints location

saveStep = 1000                   # save model every savestep default=1000



# evaluate
trainRatio = 0.8                    # train percentage
evaluateCheckpointsPath = "./checkpoints/evaluate"
```

在调试控制台中输入 python main.py -m test/head/train 可以分别开启作诗、作藏头诗、训练三种功能，以作诗为例：

```
PS C:\Users\qq185\OneDrive - bupt.edu.cn\python\AI诗人-2018211604-2018213344-凌国瀚\基于tensorflow作诗> python main.py -m test
训练样本总数:  20496
测试样本总数:  0
genrating...
WARNING:tensorflow:From C:\Users\qq185\OneDrive - bupt.edu.cn\python\AI诗人-2018211604-2018213344-凌国瀚\基于tensorflow作诗\model.py:22: BasicLSTMCell.__init__ (from tensorflow.python.ops.rnn_cell_impl) is deprecated a
nd will be removed in a future version.
Instructions for updating:
This class is deprecated, please use tf.nn.rnn_cell.LSTMCell, which supports all the feature this cell currently has. Please replace the existing code with tf.nn.rnn_cell.LSTMCell(name='basic_lstm_cell').
WARNING:tensorflow:At least two cells provided to MultiRNNCell are the same object and will share weights.
2018-12-28 19:59:24.591102: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2
restored ./checkpoints/songci\songci-2999
歆径弯浦。晴静不个、喉钗香幅。
响挽相思，轻前凤渺，
媚远携沈鳞顷。
江里山明落阿它，郊节汀楼无与中。
仿庭流。一尺上空罗案，
结蓬碑。竹得相安之我，
宝对溪儿倾薄不。
冰霄绣葵吐。晚香斜板，
华无染闭。且午玉宫天浦，
迢外黄豆，碧卷绿骊光。
况足年人析会，天宵多破苍洲。
豆水，折露香琥脂。
起问齿、归怀断。有重羞、玉辉一顷，
闲意觉清飞舞。
六叶依真甚今宜。叹桃和处指垣祥。
绿暖阑干浮辇，望阳偎工独凤帆招，
塞上不是且亏。泽断农明元年他顾。
占行剪眼风新月。
洁雪清乾，整沈泽敛，
怜空似柔浓。罗赏浇赏不今兴，
烟鳞堂上无致。飘翦桥阴，
催似无约，吹影交随。
谁度四天初也，衹是常盟误。
笑装同、料回花惬，天龙容曳。
几飞尘岁修杯卷，老谢强啧阿应事。
款断怕，月晴情，
时俗溢。最是断，
连门瘦舞。孤时调宝馥嶂国，
洗尽取人扇。
PS C:\Users\qq185\OneDrive - bupt.edu.cn\python\AI诗人-2018211604-2018213344-凌国瀚\基于tensorflow作诗>
```

此处是用已经训练的宋词模型作的诗



```python
# coding: utf-8

#evalute model, just for test
import data
from model import *

class EVALUATE_MODEL(MODEL):
    """evaluate model class"""
    def evaluate(self, reload=True):
        """evaluate model"""
        print("training...")
        gtX = tf.placeholder(tf.int32, shape=[batchSize, None])  # input
        gtY = tf.placeholder(tf.int32, shape=[batchSize, None])  # output

        logits, probs, a, b, c = self.buildModel(self.trainData.wordNum, gtX)

        targets = tf.reshape(gtY, [-1])

        # loss
        loss = tf.contrib.legacy_seq2seq.sequence_loss_by_example([logits], [targets],
                                                                  [tf.ones_like(targets, dtype=tf.float32)])
        globalStep = tf.Variable(0, trainable=False)
        addGlobalStep = globalStep.assign_add(1)

        cost = tf.reduce_mean(loss)
        trainableVariables = tf.trainable_variables()
        grads, a = tf.clip_by_global_norm(tf.gradients(cost, trainableVariables), 5) # prevent loss divergence caused by gradient explosion
        learningRate = tf.train.exponential_decay(learningRateBase, global_step=globalStep,
                                                  decay_steps=learningRateDecayStep, decay_rate=learningRateDecayRate)
        optimizer = tf.train.AdamOptimizer(learningRate)
        trainOP = optimizer.apply_gradients(zip(grads, trainableVariables))

        with tf.Session() as sess:
            sess.run(tf.global_variables_initializer())
            saver = tf.train.Saver()

            if not os.path.exists(evaluateCheckpointsPath):
                os.mkdir(evaluateCheckpointsPath)

            if reload:
```

判断模型拟合程度的代码

除此之外，还有段程序能够使得 json 格式的文本转换为该项目所需的数据格式，见 dataset 文件夹中的 jsonToTxt.py

注：该项目中也有数据库生成、词云生成和数据押韵分析，也有提取纯诗句的程序（onlypoem.py）用于生成词云，但与基于词频作诗项目大同小异，此处略去

3.（参考）诗词爬取和歌词爬取

由老师提供的代码完成的诗词与歌词的爬取，但由于歌词样本过小，同时所爬取的诗词类型混杂，下均不以此两例说明

4.另一个机器学习的模型

仍然是基于 tensorflow 的机器学习模型，但是作者给出了更为详细的代码分析

数据预处理

预处理生成 Tensor，这里笔者用大家都熟悉的《悯农》来举例分析：

```
** reading text file =
^锄禾日当午，汗滴禾下土。谁知盘中餐，粒粒皆辛苦。$
```

```
** counter = dict_items([('汗', 1), ('$', 1), ('滴', 1), ('午', 1), ('苦', 1),
('粒', 2), ('。', 2), ('土', 1), ('辛', 1), ('日', 1), ('知', 1), ('中', 1), ('皆',
1), ('当', 1), ('餐', 1), ('禾', 2), ('^', 1), ('锄', 1), (', ', 2), ('盘', 1),
('下', 1), ('谁', 1)])
```

```
** vocab_size = 23
```

```
** chars = ('粒', '。', '禾', ', ', '汗', '$', '滴', '午', '苦', '土', '辛', '日',
'知', '中', '皆', '当', '餐', '^', '锄', '盘', '下', '谁', '*')
```

```
** vocab = {'辛': 10, '滴': 6, '午': 7, '苦': 8, '谁': 21, '粒': 0, '。': 1, '土':
9, '汗': 4, '日': 11, '知': 12, '中': 13, '皆': 14, '当': 15, '餐': 16, '禾': 2,
'*': 22, '^': 17, '锄': 18, ', ': 3, '盘': 19, '下': 20, '$': 5}
```

```
** tensor = [[17, 18, 2, 11, 15, 7, 3, 4, 6, 2, 20, 9, 1, 21, 12, 19, 13, 16, 3, 0,
0, 14, 10, 8, 1, 5]]
```

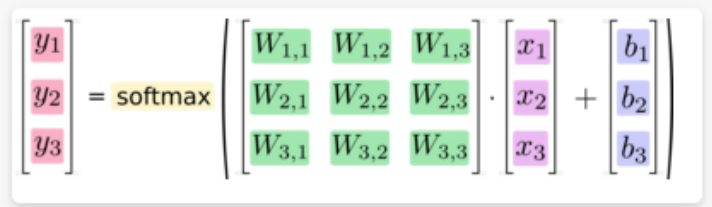
作者将诗句处理为矩阵便于程序处理

训练模型

这里笔者直接摘出核心，其实采用的是 softmax 回归模型来给不同的字符对象分配概率，如下所示：

```
self.logits = tf.matmul(output, softmax_w) + softmax_b
self.probs = tf.nn.softmax(self.logits)
```

表示成矩阵如下：



进一步写成表达式如下：

$$y = \text{softmax}(Wx + b)$$

作者称，此处采用的是 softmax 回归模型：
http://deeplearning.stanford.edu/wiki/index.php/Softmax%E5%9B%9E%E5%BD%92

但这方面已经超出了我们的学习范围

```
optional arguments:
  -h, --help            show this help message and exit
  --save_dir SAVE_DIR   directory to store checkpointed models
  --rnn_size RNN_SIZE   size of RNN hidden state
  --num_layers NUM_LAYERS
                        number of layers in the RNN
  --model MODEL         rnn, gru, or lstm
  --batch_size BATCH_SIZE
                        minibatch size
  --num_epochs NUM_EPOCHS
                        number of epochs
  --save_every SAVE_EVERY
                        save frequency
  --grad_clip GRAD_CLIP
                        clip gradients at this value
  --learning_rate LEARNING_RATE
                        learning rate
  --decay_rate DECAY_RATE
                        decay rate for rmsprop
  --init_from INIT_FROM
                        continue training from saved model at this path. Path
                        must contain files saved by previous training process:
                        'config.pkl' : configuration; 'chars_vocab.pkl' :
                        vocabulary definitions; 'iterations' : number of
                        trained iterations; 'losses-*' : train loss;
                        'checkpoint' : paths to model file(s) (created by tf).
                        Note: this file contains absolute paths, be careful
                        when moving files around; 'model.ckpt-*' : file(s)
                        with model definition (created by tf)
```

训练可选项，可在控制台通过 python trainer.py+相关可选项 来调用

```
optional arguments:
  -h, --help            show this help message and exit
  --save_dir SAVE_DIR   model directory to store checkpointed models
  --prime PRIME         输入指定文字生成藏头诗
  --sample SAMPLE       0 to use max at each timestep, 1 to sample at each
                        timestep
```

作诗可选项，可在控制台通过 python generator.py+相关可选项 来调用

```
2019-01-11 10:52:10.891686: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 6381 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1070 with Max-Q Design, pci bus id: 0000:01:00.0, compute capability: 6.1)
陶令啄文地，诗书屐齿声。晚阴须一树，前艇得薰弦。干*书高处，吟贫匪薜萝。独含荒草在，我上伴僧期。泛酒相携赋，苍龙终日求。别离空羽盖，占耳问新汀。三叶开迷径，危泉隔怕春。秋云临碧水，蕙叶下房墙。
```

生成诗句

上文：陶令啄文地，诗书屐齿声。晚阴须一树，前艇得薰弦。干*书高处，吟贫匪薜萝。独含荒草在，我上伴僧期。泛酒相携赋，苍龙终日求。别离空羽盖，占耳问新汀。三叶开迷径，危泉隔怕春。秋云临碧水，蕙叶下房墙。

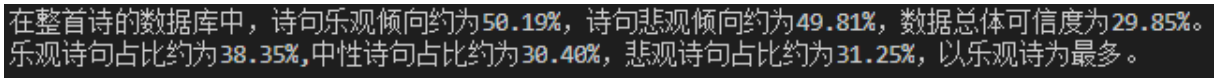
藏头诗

5.百度情感识别系统/百度 DNN 语言模型

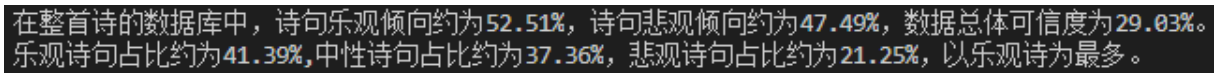
（1）百度情感识别系统

目的：对包含主观观点信息的文本进行情感极性类别（积极、消极、中性）的判断，并给出相应的置信度。

主要文件：sentiment analysis.py



基于 tensorflow 的诗词数据的情感分析



基于词频作诗的诗词数据的情感分析

（2）百度 DNN 语言模型

目的：判断一句话是否符合语言表达习惯

主要文件：dnnmode.py

基于 tensorflow 的诗词源数据的可读性分析

基于词频作诗的诗词数据的可读性分析

<3>项目意义

初步了解词频生成和机器学习两种机器自动作诗方法，并比较两者的异同，了解两者的优劣，特别是当前行业热点的机器学习方面，使得对 python 这门语言在更深层次上的应用有了更多的了解。与此同时，也能够初步了解不同算法对相同任务处理后的结果，认识到算法的重要性。

二、数据分析

<1>基于词频

该项目的数据基于 http://www.gushiwen.org 网站所爬取的几千首诗句，以及基于 tensorflow 项目中的唐诗数据，将其中的高频词提出后生成如下词云（不分词性，来源：zzcf.txt）

所采用的数据中押韵诗句与非押韵诗句比率如下：

```
Opened database successfully
rhyme rate is 97.34%
Operation done successfully
```

为 97.34%

<2>基于 tensorflow

该项目诗词数据来源：https://github.com/chinese-poetry/chinese-poetry
包含 5.5 万首唐诗、26 万首宋诗和 2.1 万首宋词
各个数据生成的词云如下（来源：poetySong.txt）：

以下为原始数据作者整理的词云数据：

唐诗高频词

宋诗高频词

宋词高频词

所采用的数据中押韵诗句与非押韵诗句比率如下：

```
Opened database successfully
rhyme rate is 94.74%
Operation done successfully
```

为 94.74%

数据准确率如下（evaluate.py）：

```
accuracy: 0.83
```

epoch 为 4

```
accuracy: 0.83
```

epoch 为 10

对于额外的 tensorflow 模型，通过 wordcloud 也统计出了其词云，如下图：

另一个 tensorflow 模型的词云

<3>总结

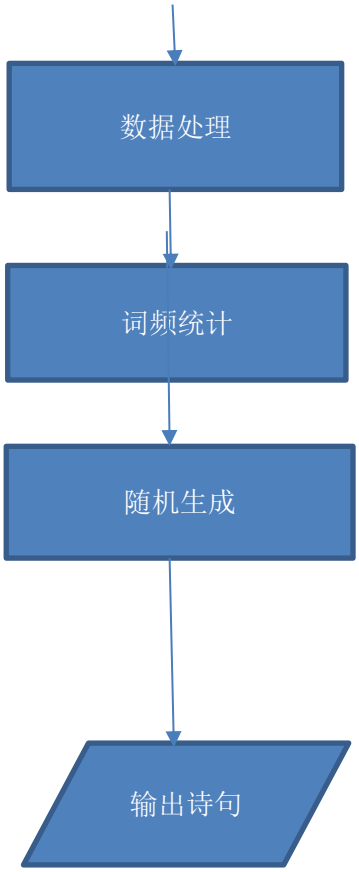
显然，机器学习作诗所需的数据量要远远高于词频作诗。因为词频作诗仅仅是提取诗词中的高频词，再按照一定规律随机生成诗句，数据量的大小和输出诗句的形式和优美程度没有相关性。
而机器学习则需要庞大的数据量来训练模型才能够确保输出的准确性，动辄上万的数据量显然要大的多。当然，数据量不能无限制的增大，否则会出现“过拟合”等现象。

而二者数据的押韵比率差别不大，可以预见其对结果的影响不如其他变量。

三、诗词生成程序的算法比较

<1>基于词频

该项目的基本思想是，在一定规则（押韵）下，在高频词中选取词语随机生成诗句。



在爬虫部分，该项目是基于 urllib3（爬取 HTML）和 BeautifulSoup4（处理 HTML）爬取古诗文网的数据，代码如下：

```python
import urllib3
from bs4 import BeautifulSoup
import certifi
import time
import random
import re

rex=re.compile(r'\(.*?\)')

f = open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/s_song.txt', "w+",encoding="utf-8")
http = urllib3.PoolManager(
    cert_reqs='CERT_REQUIRED',
    ca_certs=certifi.where())

for poemId in range(1, 1001):
    print(poemId)
    url ='https://www.gushiwen.org/shiwen/default_3A888888888888A'+str(poemId)+'.aspx'
    #'http://www.gushiwen.org/wen_' + str(poemId) + '.aspx' 原始
    #content = soup.find('div', class_="contson").get_text()
    #file.write(content)

    #'https://www.gushiwen.org/shiwen/default.aspx?page=' + str(poemId) 所有诗
    #'https://www.gushiwen.org/shiwen/default_3A666666666666A'+str(poemId)+'.aspx'   唐诗
    #'https://www.gushiwen.org/shiwen/default_3A888888888888A'+str(poemId)+'.aspx'   宋诗
    r = http.request('GET', url)
    soup = BeautifulSoup(r.data, 'html.parser')
    content = soup.find('div', class_="contson").get_text()
    content = re.sub(rex, "", content)
    content = content.replace('\n', '')
    author = soup.find('p', class_="source").get_text()
    author.replace('\n', '')
    author=author[author.find(": ")+1:]
    title = soup.find('b').get_text()
    title.replace('\n', '')
    result=(title+'::'+author+'::'+content+'\n')
    # if poemId >= 1109 and poemId < 2010:
    #     file.write(content)
    f.write(result)
    time.sleep(random.random()*3)
f.close()
```

在以上代码中，urllib3 用于爬取 HTML，BeautifulSoup4 用于处理 HTML，certifi 的作用是令爬虫使用

Mozilla 的根证书（反反爬），最后不能缺少 time 模块的停顿。这样以后才能使得数据正确的被下载。最终得到如下数据（使用原始的 url 而非例图的唐诗 url）：

卷一 卷1_1 「帝京篇十首」李世民 秦川雄帝宅，函谷壮皇居。绮殿千寻起，离宫百雉馀。 连薨遥接汉，飞观迥凌虚。云日隐层阙，风烟出绮疏。 岩廊罢机务，
欣天地康。车轨同八表，书文混四方。 赫奕俨冠盖，纷纶盛服章。羽旄飞驰道，钟鼓震岩廊。 组练辉霞色，霜戟耀朝光。晨胥怀至理，终愧抚遐荒。 卷1_5 「
浪浅深明。 斑红妆蕊树，圆青压溜荆。迹岩劳傅想，窥野访莘情。 巨川何以济，舟楫伫时英。 卷1_14「春日玄武门宴群臣」李世民 韶光开令序，淑气动芳年。
材力，终藉栋梁深。 弥怀矜乐志，更惧戒盈心。愧制劳居逸，方规十产金。 卷1_23「首春」李世民 寒随穷律变，春逐鸟声开。初风飘带柳，晚雪间花梅。 碧林

再使用正则表达式将其变为可处理的形式，代码如下：

```
import re

poemfile = open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/poem.txt', encoding='UTF-8').read()
new_poemfile = open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于tensorflow作诗/dataset/poetryTang/poetryTang.txt', encoding='UTF-8').read(

p1 = r"[\u4e00-\u9fa5]{5,7}[\u3002|\uff0c]"     #[汉字]{重复5-7次}[中文句号|中文逗号]

pattern1 = re.compile(p1)        #题目和作者
result1 = pattern1.findall(poemfile)
result2 = pattern1.findall(new_poemfile)   #搜索匹配的字符串，得到匹配列表

with open('C:/Users/qq185/OneDrive - bupt.edu.cn/python/AI诗人-2018211604-2018213344-凌国瀚/基于词频作诗/data/zzcf.txt','a+',encoding='utf-8') as f: #打开输出文件
    for x in result1:
        f.write(x)
    for y in result2:
        f.write(x)
poemfile.close()
new_poemfile.close()
```

此处额外使用了 tensorflow 项目的数据

处理后文件格式为：

秦川雄帝宅，函谷壮皇居。绮殿千寻起，离宫百雉馀。连薨遥接汉，飞观迥凌虚。云日隐层阙，风烟出绮疏。岩廊罢机务，崇文聊驻辇。玉匣启龙图，金绳披凤
定，怀柔万国夷。梯山咸入款，驾海亦来思。单于陪武帐，端扆朝四岳，无为任百司。霜节明秋景，轻冰结水湄。芸黄遍原隰，禾颖积京畿。共乐还乡宴，代马
烧火红。雕戈夏服箭，羽骑绿沉弓。怖兽潜幽壑，惊禽散翠空。长烟晦落景，灌木振严风。所为除民瘼，非是悦林丛。烈烈寒风起，惨惨飞云浮。霜浓凝广隰，
已获千箱庆，何以继熏风。碧昏朝合雾，丹卷暝韬霞。结叶繁云色，凝琼遍雪华。光楼皎若粉，映幕集疑沙。泛柳飞飞絮，妆梅片片花。照壁台圆月，飘珠箔穿

接下来使用 jieba 模块提取各个词性的高频词，数据每 10 个换一行

```
count = 0
with open(n_file,'w+',encoding='utf-8') as f1:
    for x in jieba.analyse.textrank(content, topK=600, allowPOS=('n', 'nr', 'ns', 'nt', 'nz', 'm')):
        f1.writelines(x+" ")
        count+=1
        if count%10==0:
            f1.write("\n")
```

以名词为例

天地 芳草 行人 烟霞 东风 芙蓉 风雨 十年 洞庭 萧萧
白发 天涯 杨柳 清风 江南 千年 风流 青云 桃花 白头

数据如上

最后是诗句的生成，通过 pinyin 模块控制平仄和韵脚，再以“名名动名名”的形式输出诗句

```
def Line5():
    noun = open('D:/data/freqword_n.txt', encoding='utf-8').readlines()
    verb = open('D:/data/freqword_v.txt', encoding='utf-8').readlines()

    nounlist = []
    for word in noun:
        # outfile.write(pinyin.get(word, format="strip")+" ")
        i = 0
        while i<len(word):
            if word[i:i+2]!="\n":
                nounlist.append(word[i:i+2])
            i=i+3

    verblist = []
    for word in verb:
        i = 0
        while i<len(word):
            if word[i:i+2]!="\n":
                verblist.append(word[i:i+2])
            i=i+3
```

储存名词和动词

```
num = 0

rhythm = ""
rhythmList = [['a', 'ua', 'ia'], ['e', 'o', 'uo'], ['e', 'ie', 've'], ['i', 'v', 'er'], ['u'], ['ai', 'uai'], ['ei', 'ui'], ['ao', 'iao'], [
    'ou', 'iu'], ['an', 'ian', 'uan', 'van'], ['en', 'in', 'un', 'vn'], ['ang', 'iang', 'uang'], ['eng', 'ing', 'ueng', 'ong', 'iong']] # 汉字十三辙
while num < 4:  # 诗句数量可调整为8句绝句
    i = random.randint(1, len(nounlist) - 1)
    i1 = random.randint(1, len(nounlist) - 1)
    j = random.randint(1, len(verblist) - 1)

    ind = 0
    ind1 = 0
    if num == 0:

        #首句用绝句的仄起仄收式：仄仄平平仄
        a = [[0],[0]]
        b = [5]
        c = [[5],[0]]
        while int(a[0][-1])<3 or int(a[1][-1])<3 or int(b[-1])>2 or int(c[0][-1])>2 or int(c[1][-1])<3:
            i = random.randint(1, len(nounlist) - 1)
            i1 = random.randint(1, len(nounlist) - 1)
            j = random.randint(1, len(verblist) - 1)
            a = pinyin.get(nounlist[i], delimiter=" ", format="numerical").split()
            b = pinyin.get(verblist[j][1], delimiter=" ", format="numerical")
            c = pinyin.get(nounlist[i1], delimiter=" ", format="numerical").split()
```

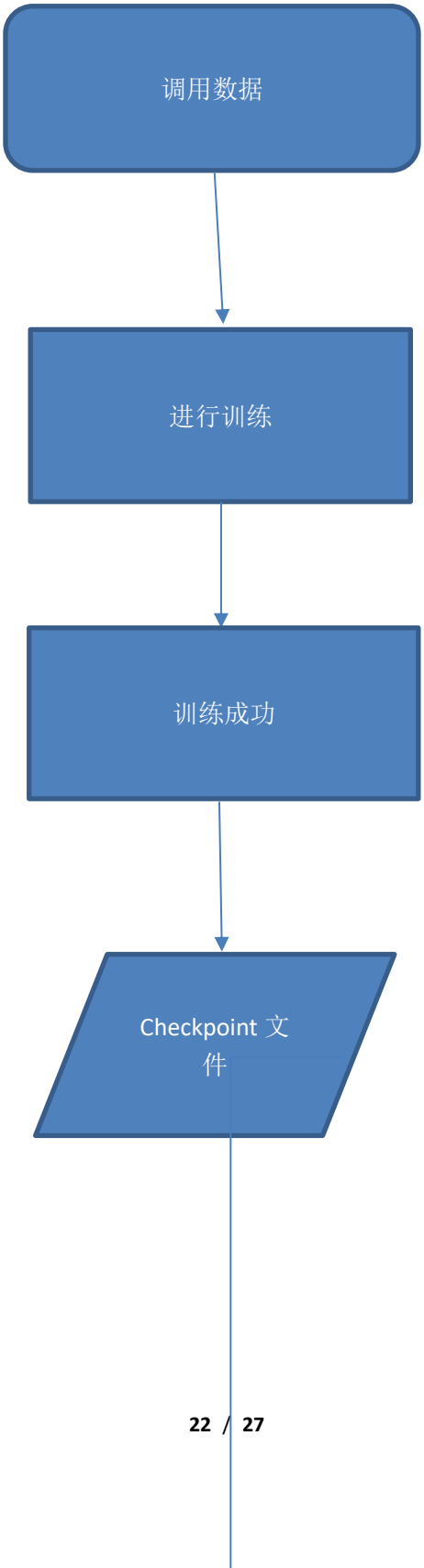
所有诗句都使用 random 模块初始化，再利用 pinyin 库限制生成诗句的平仄关系

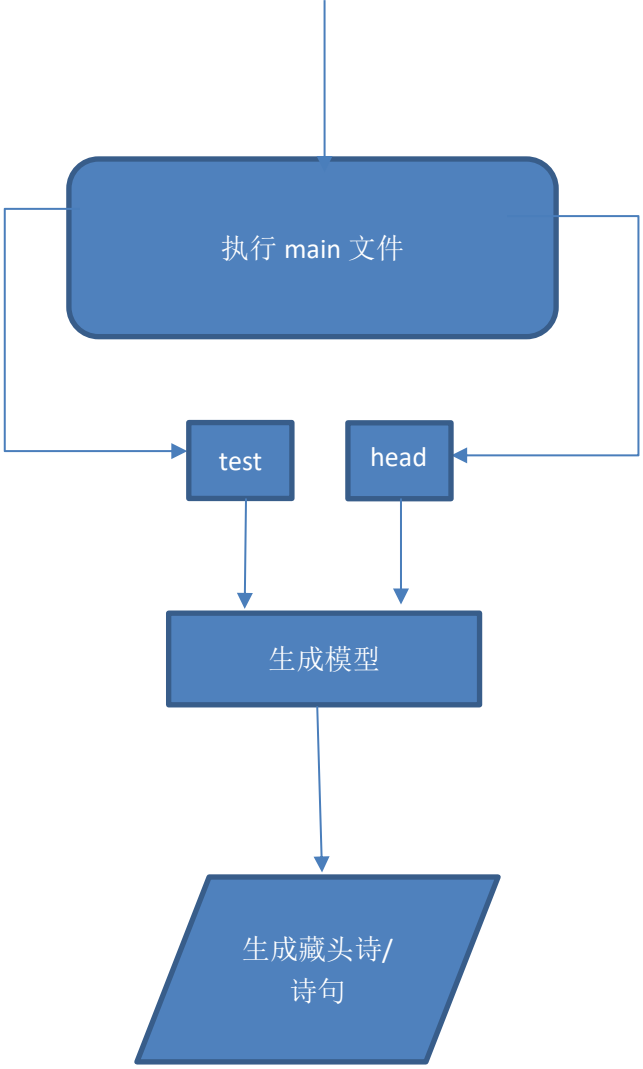
```
            #以下是押韵的处理
            rhythm = ""
            verse = pinyin.get(nounlist[i1][1], format="strip")
            for p in range(len(verse)):
                if search(rhythmList, verse[p:]):
                    ind = p
            rhythm = verse[ind:]

        if num != 0:
            ind1 = 0
            verse1 = pinyin.get(nounlist[i1][1], format="strip")
            for p in range(len(verse1)):
                if search(rhythmList, verse1[p:]):
                    ind1 = p

            while verse1[ind1:] != rhythm:
                i1 = random.randint(1, len(nounlist) - 1)
                verse1 = pinyin.get(nounlist[i1][1], format="strip")
                for p in range(len(verse1)):
                    if search(rhythmList, verse1[p:]):
                        ind1 = p

        print(nounlist[i] + verblist[j][1] + nounlist[i1])      #可通过修改此处变为律诗
        num += 1
```

此处处理韵脚的押韵关系，保证每句诗都能押韵，最后以名名动名名的形式输出

在以上代码中，也可以更改 num 的限制条件，改变诗句的长度，诗的句数以及哪几句诗句押韵。

<2>基于 tensorflow





该项目的基本思想是，通过机器学习，让机器“学会”写诗。通过生成的模型，能够写出与数据相似的诗句。

可操作和调整的主要代码在 config.py 中（主要是 train 方面）

```
import tensorflow as tf
import numpy as np
import argparse
import os
import random
import time
import collections

batchSize = 64

learningRateBase = 0.001
learningRateDecayStep = 1000
learningRateDecayRate = 0.95

epochNum = 10                       # train epoch
generateNum = 5                      # number of generated poems per time

type = "songci"                      # dataset to use, shijing, songci, etc
trainPoems = "./dataset/" + type + "/" + type + ".txt" # training file location
checkpointsPath = "./checkpoints/" + type # checkpoints location

saveStep = 1000                      # save model every savestep



# evaluate
trainRatio = 0.8                       # train percentage
evaluateCheckpointsPath = "./checkpoints/evaluate"
```

该文件可以调整数据的大小，批次，拟合数据，训练 epoc 数，以及在训练到何时保存数据

```
def __init__(self, filename, isEvaluate=False):
    """pretreatment"""
    poems = []
    file = open(filename, "r",encoding='UTF-8')
    for line in file:  #every line is a poem
        title, author, poem = line.strip().split("::")  #get title and poem
        poem = poem.replace(' ','')
        if len(poem) < 10 or len(poem) > 512:  #filter poem
            continue
        if '_' in poem or '《' in poem or '[' in poem or '(' in poem or '（' in poem:
            continue
        poem = '[' + poem + ']' #add start and end signs
        poems.append(poem)
```

处理 dataset 里的诗词数据，将题目和内容分离，然后清洗过滤掉一些不好的训练样本，包含特殊符号、字数太少或太多的都去除，最后在诗的前后分别加上开始和结束符号（方括号）

```
#counting words
wordFreq = collections.Counter()
for poem in poems:
    wordFreq.update(poem)

wordFreq[" "] = -1
wordPairs = sorted(wordFreq.items(), key = lambda x: -x[1])
self.words, freq = zip(*wordPairs)
self.wordNum = len(self.words)

self.wordToID = dict(zip(self.words, range(self.wordNum))) #word to ID
poemsVector = [([self.wordToID[word] for word in poem]) for poem in poems] # poem to vector
if isEvaluate: #evaluating need divide dataset into test set and train set
    self.trainVector = poemsVector[:int(len(poemsVector) * trainRatio)]
    self.testVector = poemsVector[int(len(poemsVector) * trainRatio):]
else:
    self.trainVector = poemsVector
    self.testVector = []
print("训练样本总数: %d" % len(self.trainVector))
print("测试样本总数: %d" % len(self.testVector))
```

统计样本总数

```
def generateBatch(self, isTrain=True):
    #padding length to batchMaxLength
    if isTrain:
        poemsVector = self.trainVector
    else:
        poemsVector = self.testVector

    random.shuffle(poemsVector)
    batchNum = (len(poemsVector) - 1) // batchSize
    X = []
    Y = []
    #create batch
    for i in range(batchNum):
        batch = poemsVector[i * batchSize: (i + 1) * batchSize]
        maxLength = max([len(vector) for vector in batch])
        temp = np.full((batchSize, maxLength), self.wordToID[" "], np.int32) # padding space
        for j in range(batchSize):
            temp[j, :len(batch[j])] = batch[j]
        X.append(temp)
        temp2 = np.copy(temp) #copy!!!!!!
        temp2[:, :-1] = temp[:, 1:]
        Y.append(temp2)
    return X, Y
```

因为诗的长度不一致，需要用空格填补，以便于模型学习

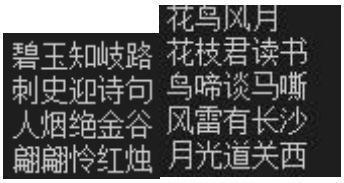
```python
def buildModel(self, wordNum, gtX, hidden_units = 128, layers = 2):
    """build rnn"""
    with tf.variable_scope("embedding"): #embedding
        embedding = tf.get_variable("embedding", [wordNum, hidden_units], dtype = tf.float32)
        inputbatch = tf.nn.embedding_lookup(embedding, gtX)

    basicCell = tf.contrib.rnn.BasicLSTMCell(hidden_units, state_is_tuple = True)
    stackCell = tf.contrib.rnn.MultiRNNCell([basicCell] * layers)
    initState = stackCell.zero_state(np.shape(gtX)[0], tf.float32)
    outputs, finalState = tf.nn.dynamic_rnn(stackCell, inputbatch, initial_state = initState)
    outputs = tf.reshape(outputs, [-1, hidden_units])

    with tf.variable_scope("softmax"):
        w = tf.get_variable("w", [hidden_units, wordNum])
        b = tf.get_variable("b", [wordNum])
        logits = tf.matmul(outputs, w) + b
```

搭建 LSTM 模型

```python
def probsToWord(self, weights, words):
    """probs to word"""
    prefixSum = np.cumsum(weights) #prefix sum
    ratio = np.random.rand(1)
    index = np.searchsorted(prefixSum, ratio * prefixSum[-1]) # large margin has high possibility to be sampled
    return words[index[0]]
```

该方法（probsToWord）引入了一些随机性，保证每次写的诗都不一样

四、实验结果分析

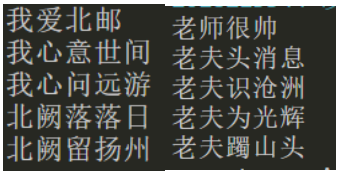


此两图为词频作诗结果

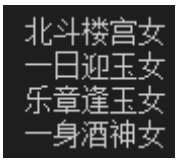
（以下分析以非藏头诗为例）

可以看到词频作诗的结果在格式上没有错误，也的确做到了第二句和第四句的押韵（“二”与“儿”），同时第一句诗还能做到“仄仄平平仄”的仄起仄收式的开头。而在诗词本身的意义方面，其百度 DNN 模型得分为 34217.8

而且更值得一提的是，如果在词频作诗中的藏头诗的诗头不为四个名词，那么往往不能正确的作诗。（如下图）



并且，很多时候诗的最后一个字也会发生重复



此两图为机器学习作诗结果

反观机器学习的作诗结果，格式正确，但是韵脚的押韵却没能做到，不过在平仄关系上却表现尚佳。

平仄关系为：

平平平仄仄平平，平仄平平平平仄。仄仄平平仄仄仄，平平平仄平平平。

可以明显的看出，平仄关系较词频作诗有了长足进步（特别是第一句和第四句）

其百度 DNN 模型得分为 4952.1

那么为什么会出现这样的原因呢？

在二、数据分析中已经提到，数据大小就不会影响词频作诗的准确率和可读性，即使是我们仅仅给程序几个关键词它也能作诗。很显然，词频作诗的一大优势就是其基于算法的稳定性，一旦算法要求哪几处押韵或平仄，该程序也会严格执行。但也正是固定的算法制约了它。可以说，只要不改变生成诗句的算法，基于词频生成的诗句几乎永远也不会拥有完整的平仄关系等诗的基本属性，甚至词性也会被限制，更不用说语义了（ppl=34217.8）。与此同时，基于高频关键字写诗也说明了作诗时使用的数据实际上不是爬取的诗，而仅仅是数个高频词。在如此小的数据量下写藏头诗，面对用户层出不穷的输入，程序也不免黔驴技穷，这就是藏头诗诗头和押韵诗韵脚出现雷同的原因。

而机器学习则完全不同。通过对相当数量的诗句的学习，该程序能写出越来越像人类诗人的诗句，在这个过程中，平仄关系和押韵关系也慢慢的被“学习”。可以预测的是，在样本量达到一定数量以后，甚至连如何写富有语义的诗句的能力也能被“学习”下来（ppl=4952.1）。然而在样本量或训练量不足的情况下，想要机器学习作诗也遵循严格的平仄和押韵还是勉为其难了。这主要是因为其在数据量和训练量不足时缺乏算法的约束。

因此，对于哪种作诗算法好，我们就可以粗略的得出结论：数据量小时，基于词频的算法作诗质量高；数据量大时，基于机器学习的算法作诗质量高。当然，当提供的数据不全是同一种类型的诗的时候（如唐诗+宋词）机器学习就有较大的劣势了。总之，对算法的输出结果影响最大的还是其使用的原始数据。