必不应

——RNN写歌词，爬虫，倒排索引
(七零八落的凑成的期末作业)

陈伟文 徐洪义 盛俊杰

Quick Start

数据获取

歌手List

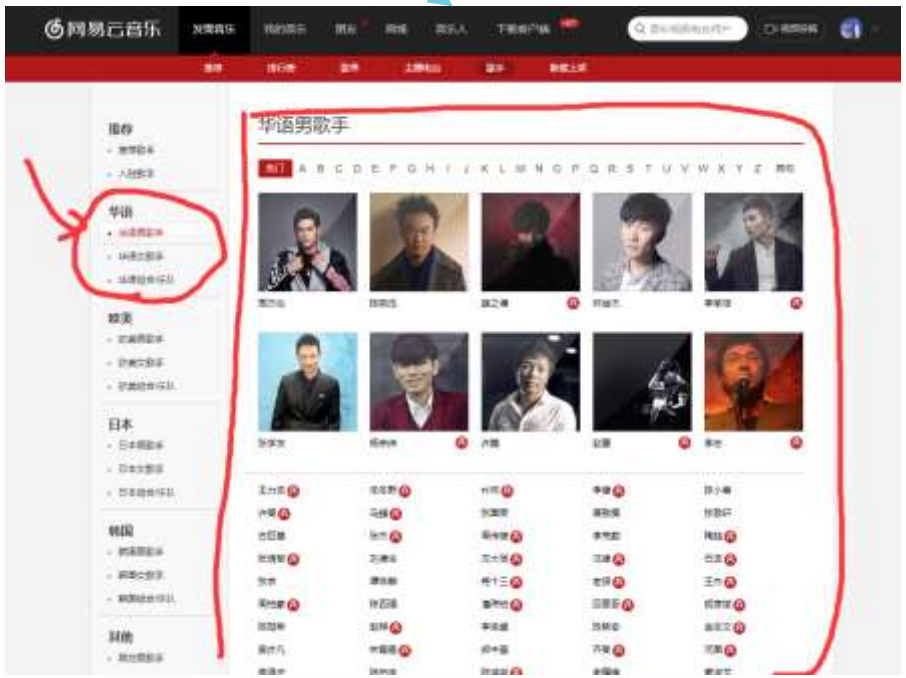
每个歌手top50的歌

歌词

封面图

外链播放器

Web展示页



和你在一起2013版[Live]

和你在一起2013版[Live] - 李志 - 00:00

昨天在梦里

我又看见你

宝贝 他们说 我不爱你

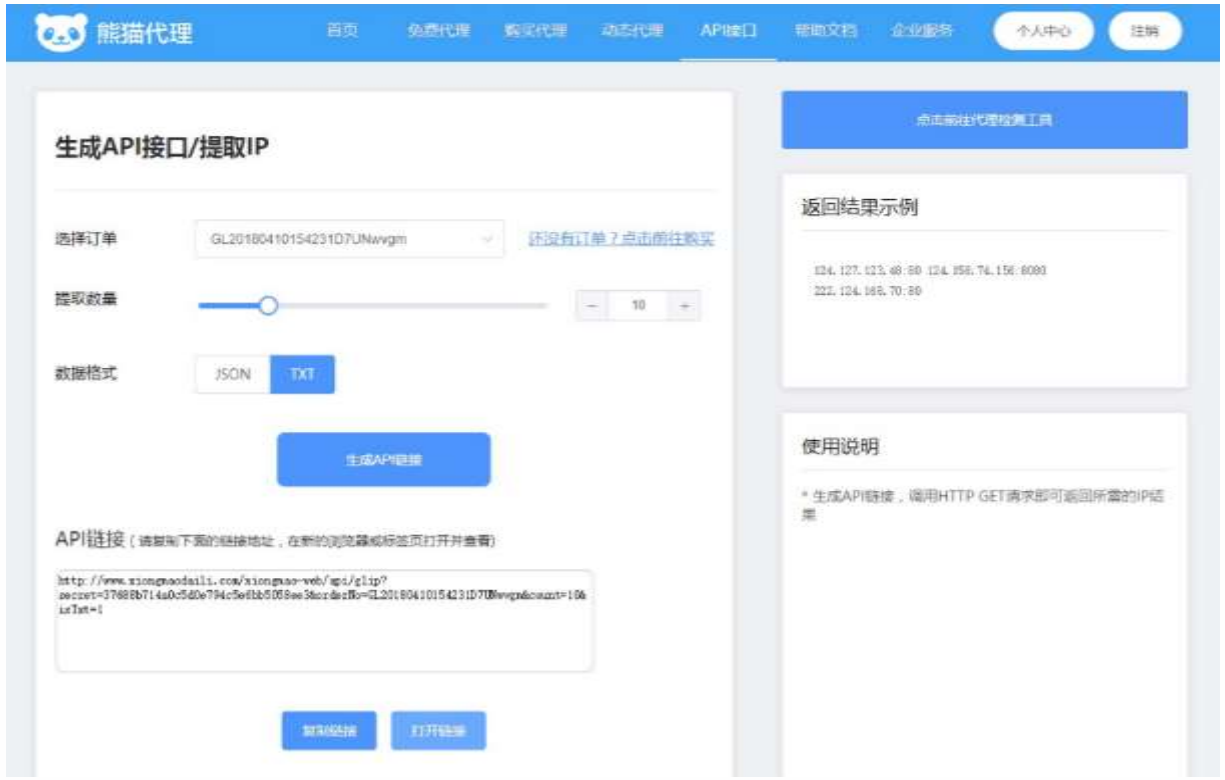
你拥有我的

不只是今夜

可是 你比我 小了六岁

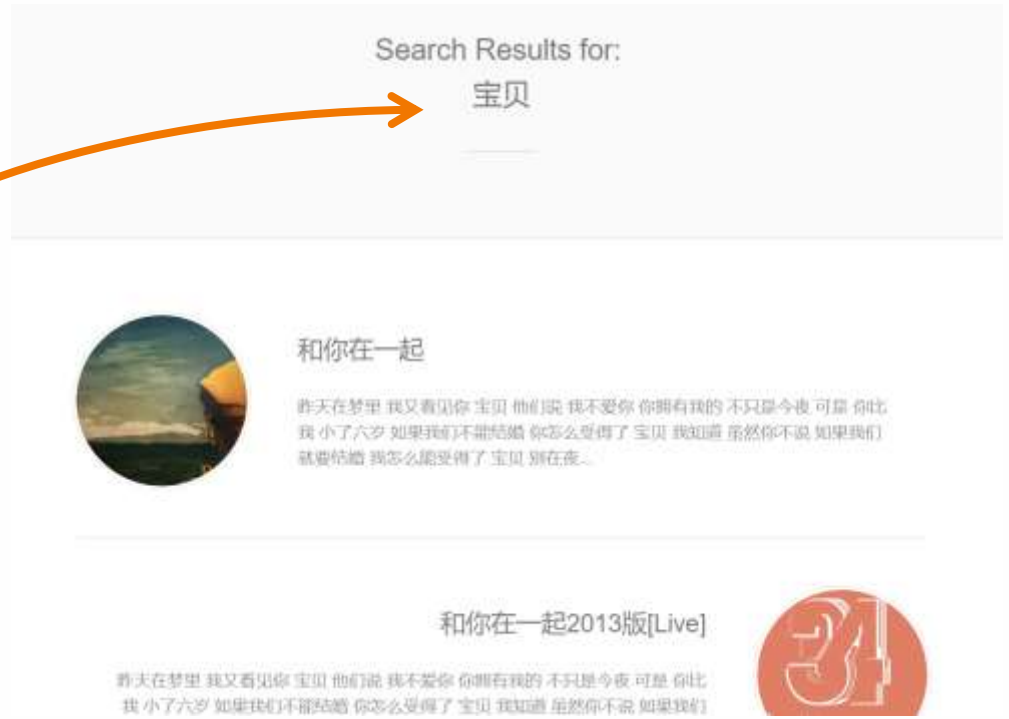
如果我们不能结婚

爬个十分钟ip就被网易云封了
不得已搞个ip池
不过换其他的ip一下子速度就降了不少



爬虫
常规
操作

歌词 → 中文分词 → 倒排索引 → 简易搜索引擎



那个，web用的python的框架
倒排索引用的java的包
然后跨语言调用就……你懂我意思吧

第二次作业做爬虫，做过了又不好意思什么都不做

Lists for WordClouds

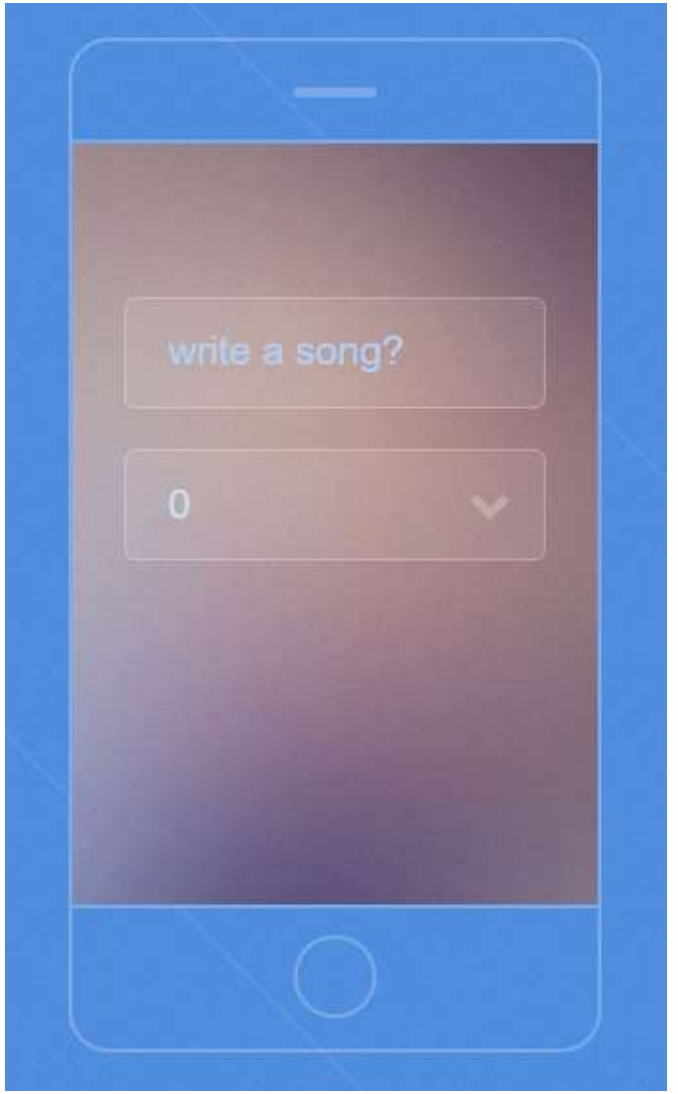
A-Lin	at17	Beyond
BOBO	By2	C-BLOCK
eliaqsopa	EXO-M	F.I.R.
F4	G.E.M.邓紫棋	GALA
HITA	Hush!	JS
MC Hotdog	Mr.	my little airport
Pianoboy高至豪	Robynn & Kendy	S.H.E

张震岳

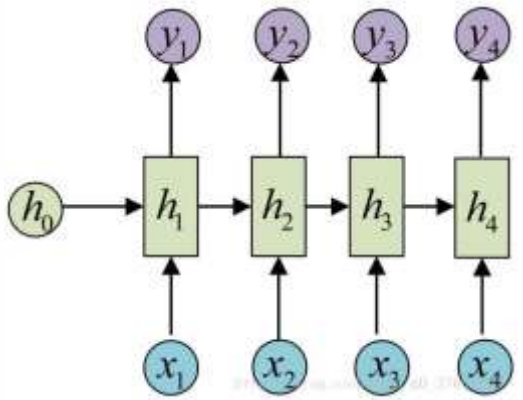
同一歌手的所有歌词 → Jieba分词 → Word Cloud

第三次作业刚了两周CCIR然后弃赛了

回到Bibuying,
把主页上的小手机玩一玩吧



2.利用RNN手动实现Char-RNN
自动歌词生成器

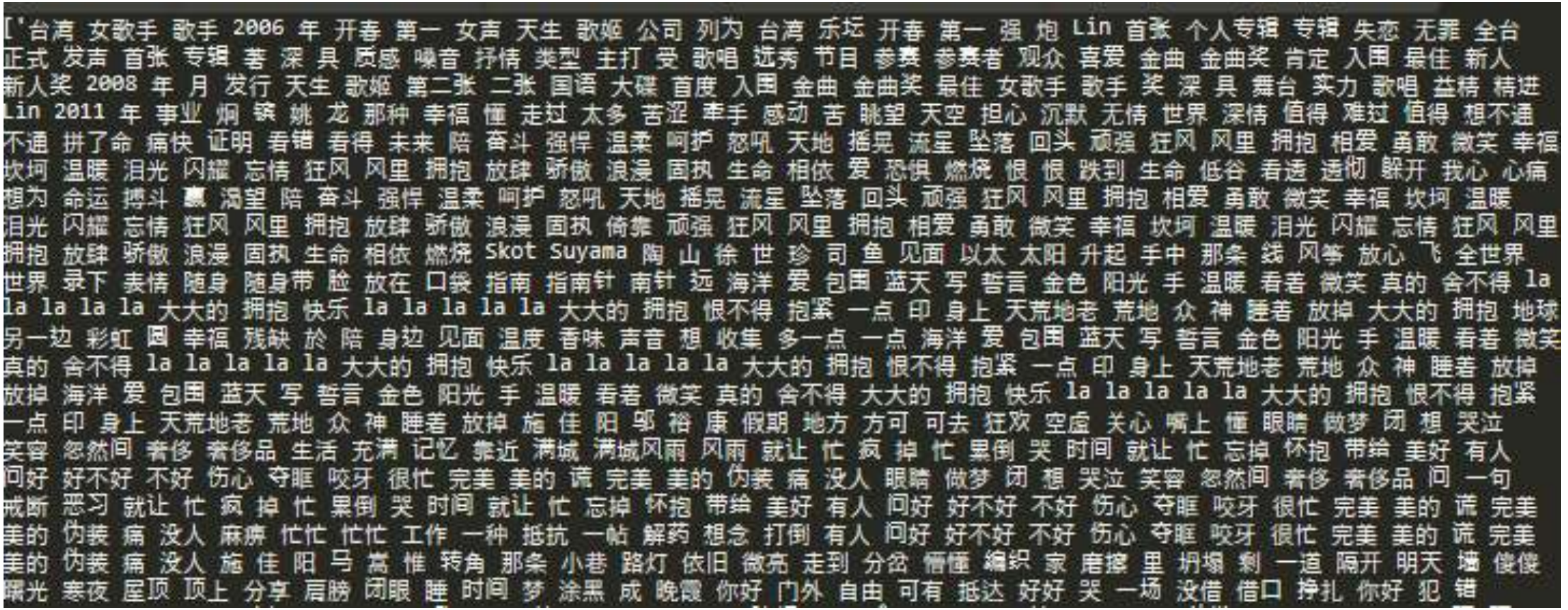


（1）歌手风格聚类

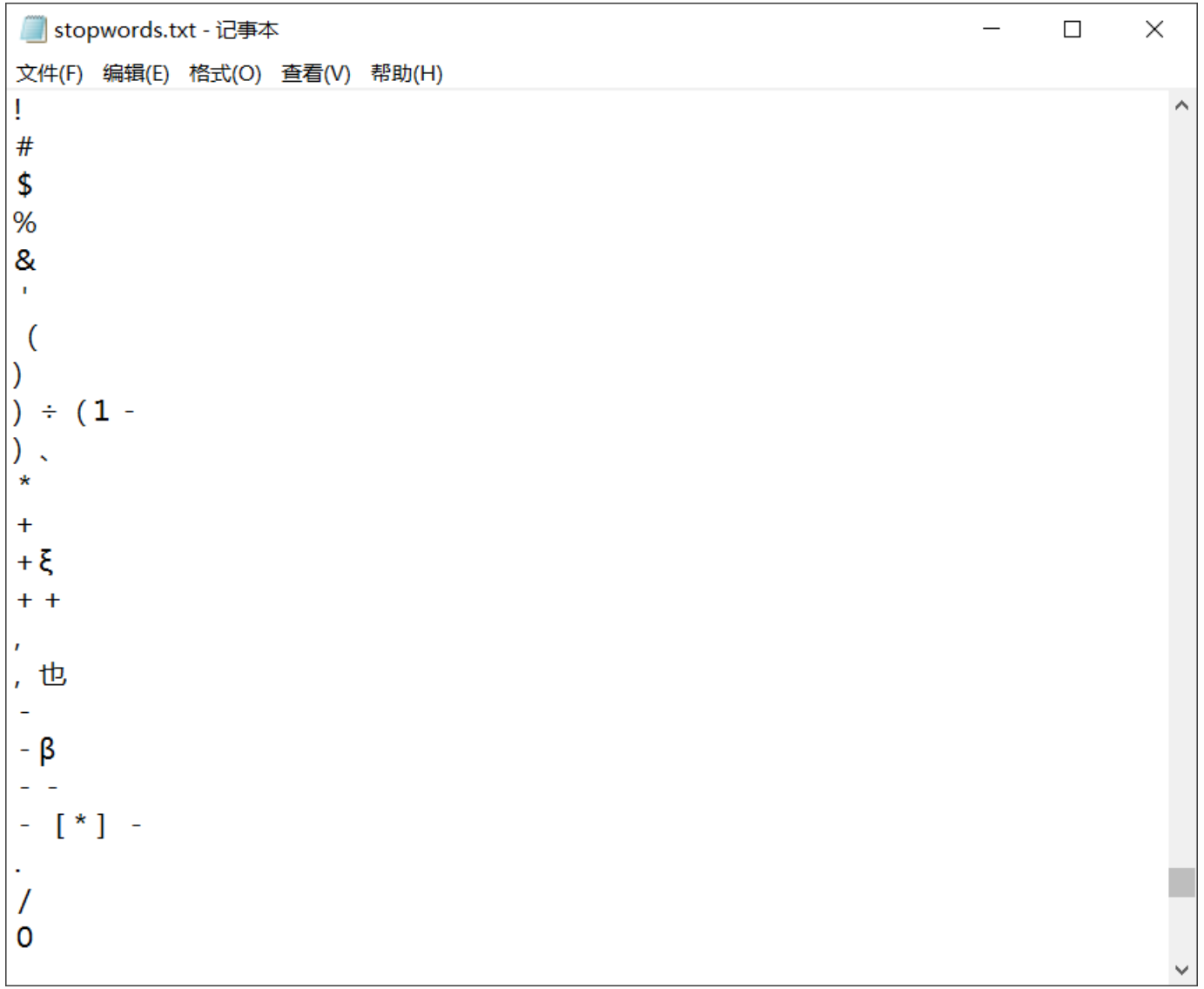
网易云300位歌手聚类

切词

这里中文切词使用结巴切词：



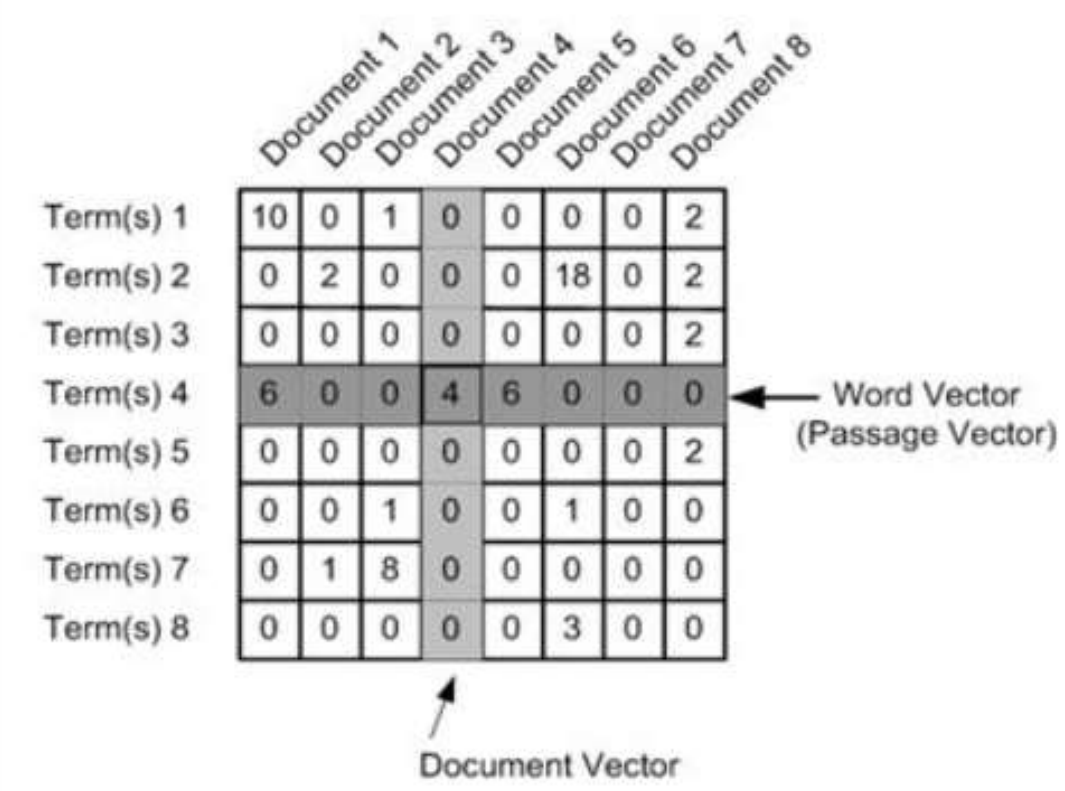
去除停用词（1893个）



TF-IDF构建词权重

使用PCA降维

将原始6w+维度降到100维




```
#开始建立tf-idf矩阵


    vectorizer = CountVectorizer()
    transformer = TfidfTransformer()
    #得到词频矩阵
    freqMatrix=vectorizer.fit_transform(word_lst)
    print("得到词频矩阵",freqMatrix.shape)
    print(freqMatrix,file=f5)

    #pca降维
    PCAfreqMatrix = TruncatedSVD(n_components=100).fit_transform(freqMatrix)
    print("pca降维",PCAfreqMatrix.shape)
    #对PCAfreqMatrix计算出tf-idf矩阵
    retfidf = transformer.fit_transform(PCAfreqMatrix)
    print("计算出tf-idf矩阵",retfidf.shape)
    print(retfidf,file=f4)
    tfidf_train=retfidf.toarray()
    print("tfidf_train.shape: ",tfidf_train.shape)
```

K-means算法聚类

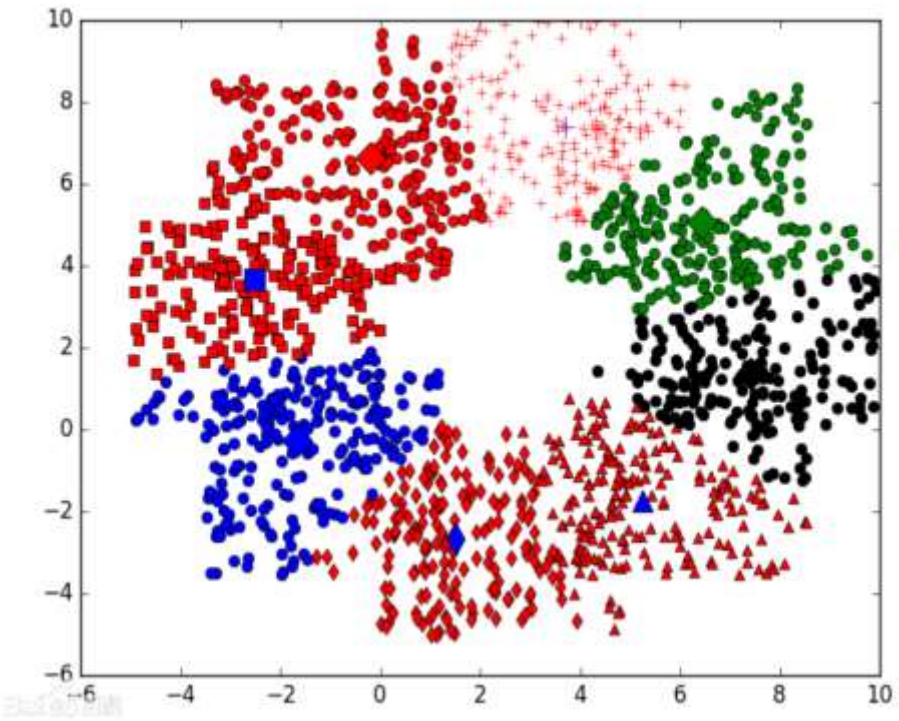
将300位歌手聚成10类：

1：初始化K个样本作为初始聚类中心；

2：计算每个样本点到K个中心的距离，选择最近的中心作为其分类，直到所有样本点分类完毕；

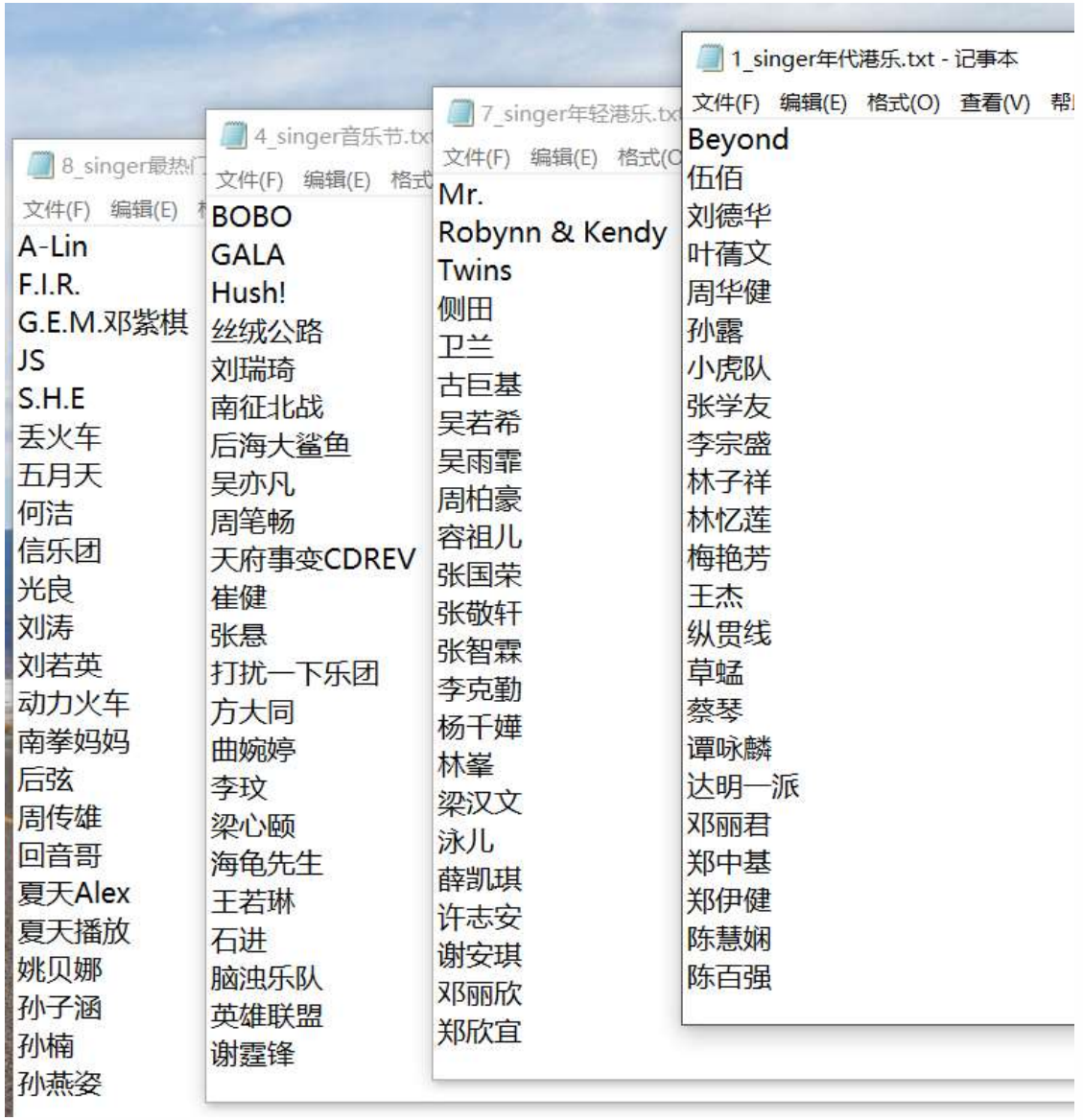
3：分别计算K个类中所有样本的质心，作为新的中心点，完成一轮迭代。

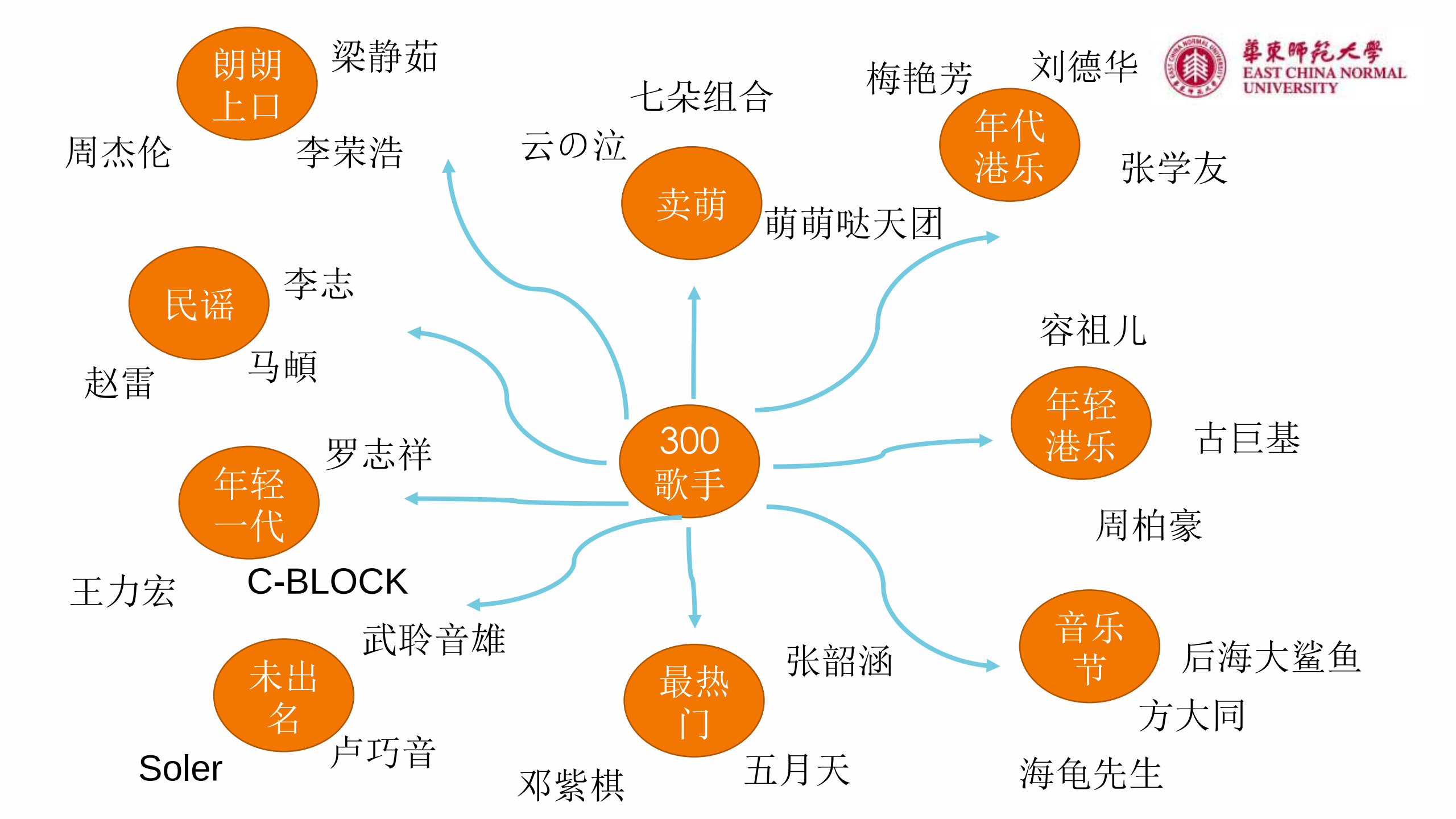
通常的迭代结束条件为新的质心与之前的质心偏移值小于一个给定阈值。



聚类结果

[7262755277228271625275227428948043555
5877972772462370773274622664006674362
7387740534574774964854929977766707662
9377127777183054122077852746741977037
2767576326247063645443865072372760532
9083418744577709222940276282792774364
4677762964945496270467976666649279373
7987202276662667627352145824800090673
457542]



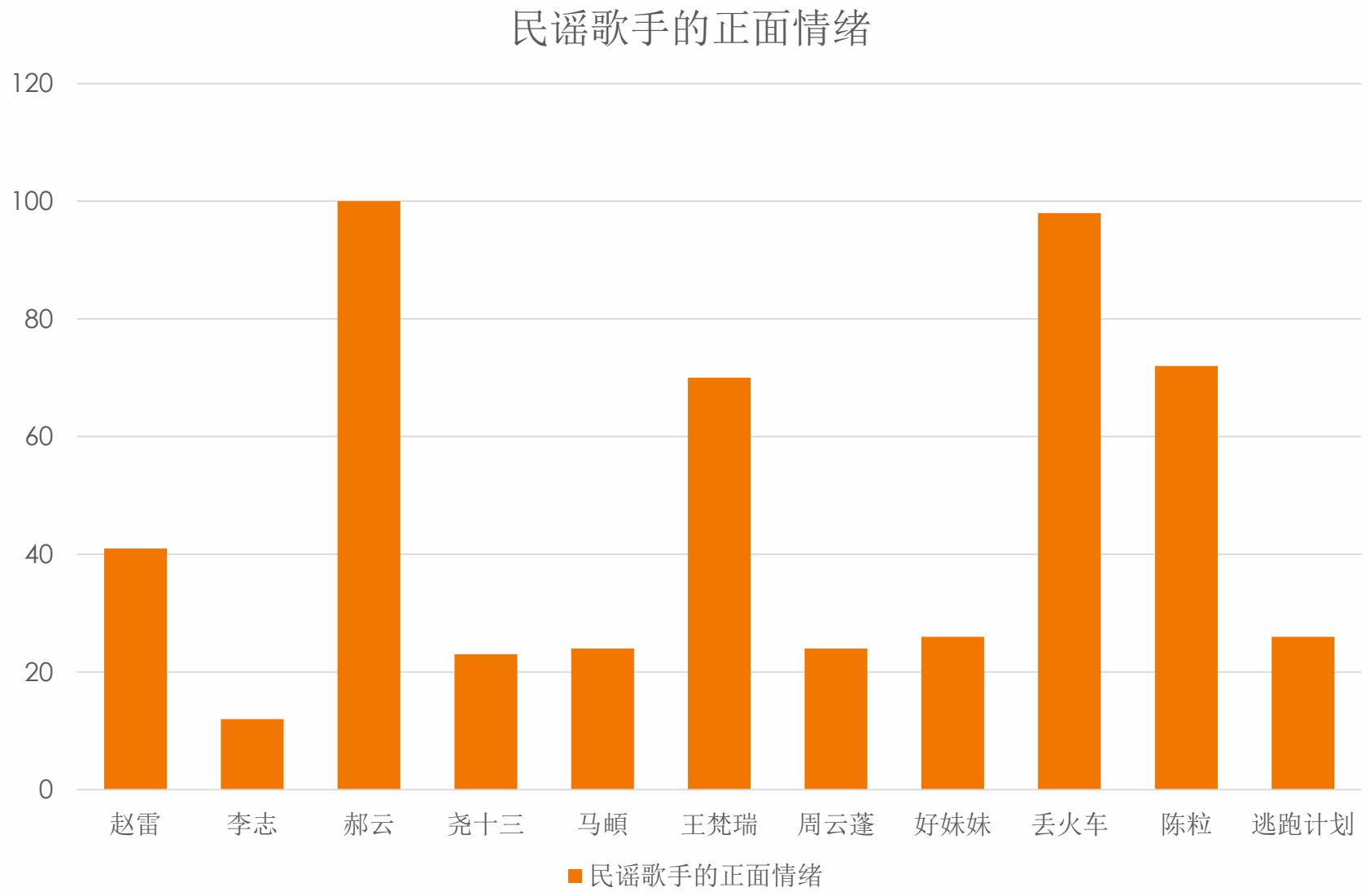

朗朗上口
梁静茹
周杰伦
李荣浩
七朵组合
云の泣
卖萌
萌萌哒天团
梅艳芳
刘德华
年代港乐
张学友
民谣
李志
赵雷
马頔
容祖儿
年轻港乐
古巨基
罗志祥
年轻一代
王力宏
C-BLOCK
300歌手
周柏豪
武聆音雄
未出名
卢巧音
Soler
张韶涵
最热门
邓紫棋
五月天
音乐节
后海大鲨鱼
方大同
海龟先生

```
def index(request):
    category = ({'idx': 0, 'name': '民族'},
                {'idx': 1, 'name': '年代港乐'},
                {'idx': 2, 'name': '未出名'},
                {'idx': 3, 'name': '年轻一代'},
                {'idx': 4, 'name': '音乐节'},
                {'idx': 5, 'name': '民谣'},
                {'idx': 6, 'name': '朗朗上口'},
                {'idx': 7, 'name': '年轻港乐'},
                {'idx': 8, 'name': '最热门'},
                {'idx': 9, 'name': '卖萌'},)
```

词频分析

```
'走': 306, '想': 291, '中': 289, '生活': 252, '永远': 241, '姑娘': 215, '我要': 205, '唱':
'走': 95, '回忆': 71, '听': 61, '太': 56, '笑': 54, '时间': 51, '雨': 50, '中': 48, '飘':
, '心': 708, '中': 689, '一生': 559, '梦': 520, '心中': 496, '里': 486, '永远': 469, '走':
'想': 286, 'La': 282, 'I': 218, 's': 178, '里': 177, 'the': 171, 'you': 166, '太': 157,
, '想': 1653, 'I': 1421, '': 1306, '': 1254, 'you': 1089, '': 884, '': 850, '': 780,
5, 'the': 2863, 'to': 1885, 'me': 1479, '爱': 1362, 'a': 1311, 'it': 1254, 'my': 1058, 's
, '': 1284, '想': 1196, '': 1108, '世界': 979, '': 882, '': 862, '里': 803, '走': 798,
'里': 245, '世界': 229, '走': 224, '时间': 202, '制作': 194, '录音室': 179, '太': 178, '中'
5, '太': 835, '里': 642, '未': 601, '中': 573, '走': 569, '做': 534, 'I': 528, 'you': 521,
5, '走': 2155, '世界': 2056, '心': 1960, '爱情': 1734, '幸福': 1676, '永远': 1628, '快乐':
```

情感分析


民谣歌手的正面情绪
120
100
80
60
40
20
0
赵雷
李志
郝云
尧十三
马頔
王梵瑞
周云蓬
好妹妹
丢火车
陈粒
逃跑计划
民谣歌手的正面情绪

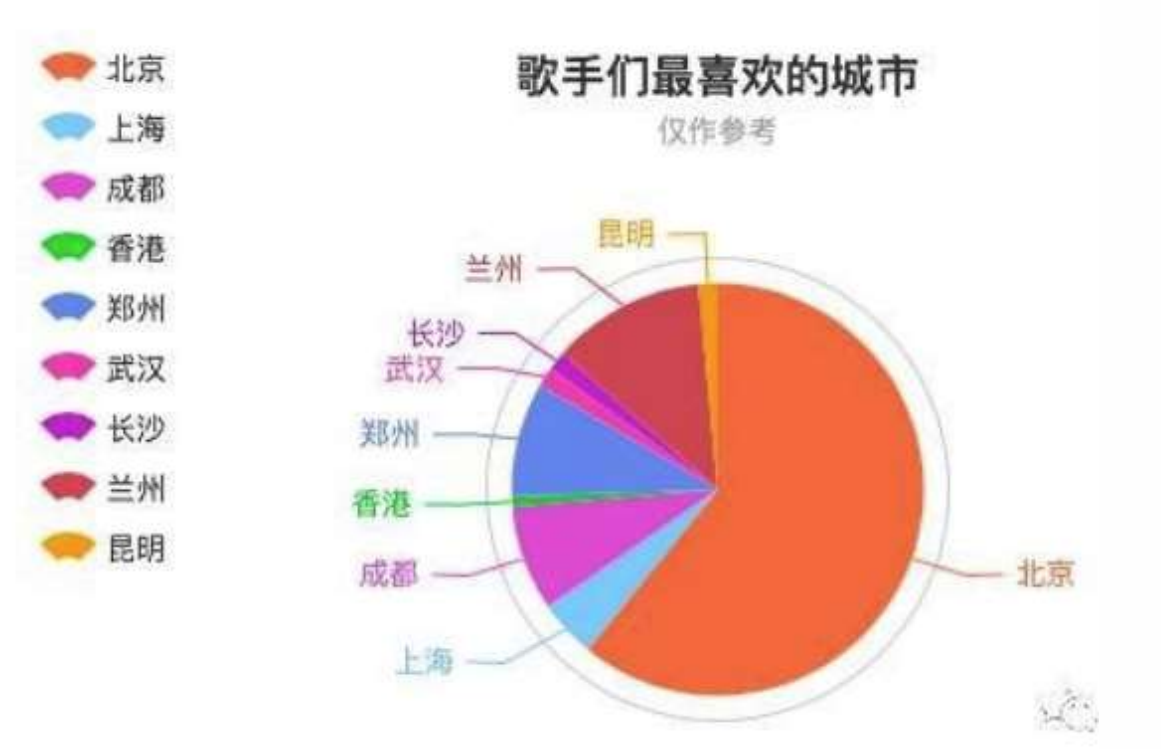
最低的是李志

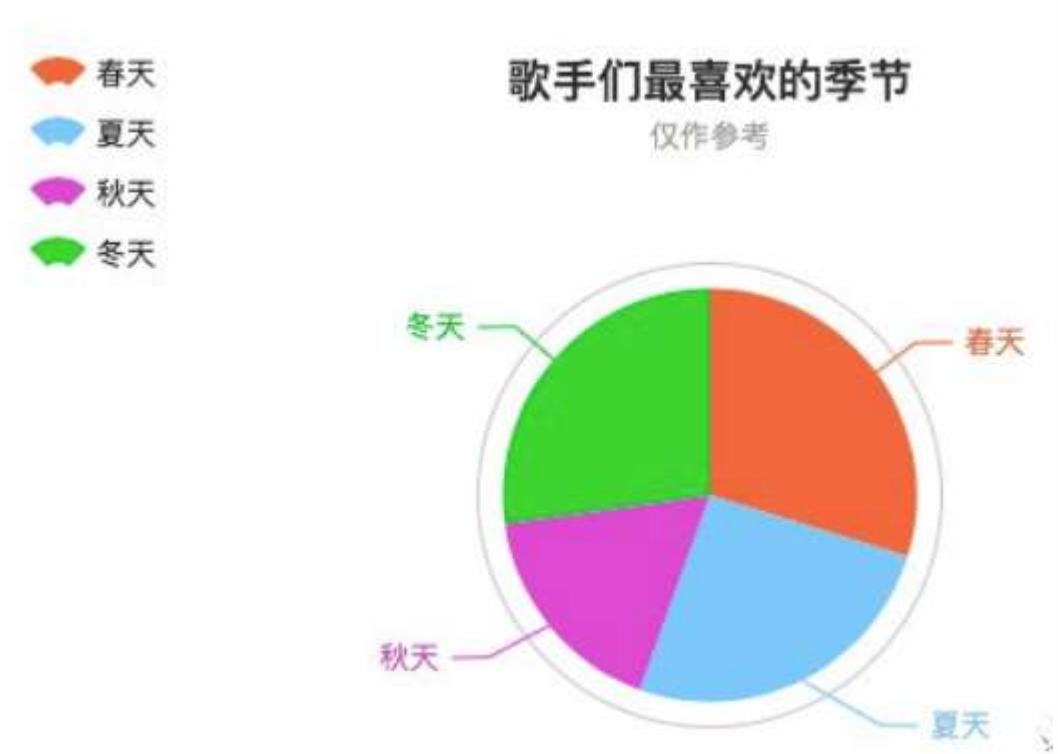
去处一部分停用词后

这让人心慌
这让人心慌

心慌：0.2478
我 ：0.2292
想起：0.2044
沉默：0.1737
宝贝：0.1511
禁忌：0.1427
孤独：0.1427

再做一些无聊的统计：城市、季节



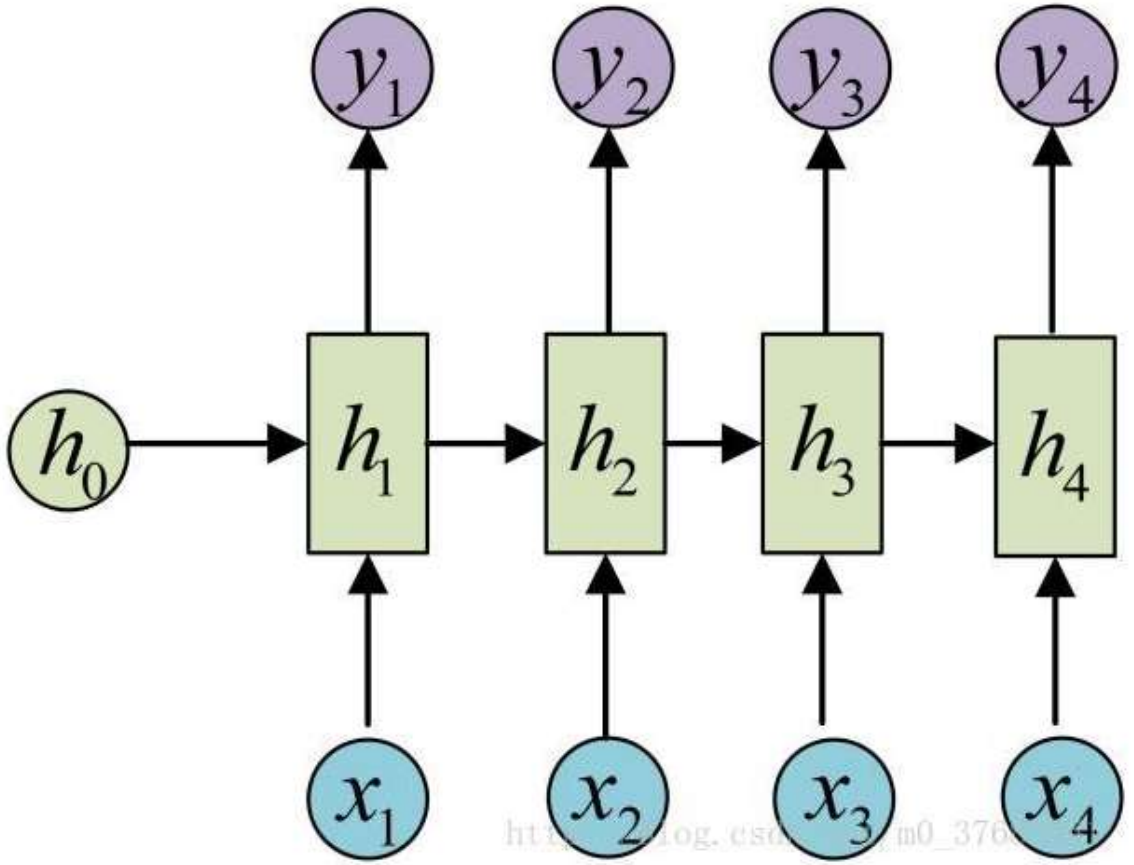


（2）实现Char-RNN歌词生成

循环神经网络 RNN

对于序列问题的建模，RNN引入了隐状态h（hidden state），h可以对序列形的数据提取特征，接着再转换为输出

$$h(t)=\sigma(z(t))=\sigma(Ux(t)+Wh(t-1)+b)$$

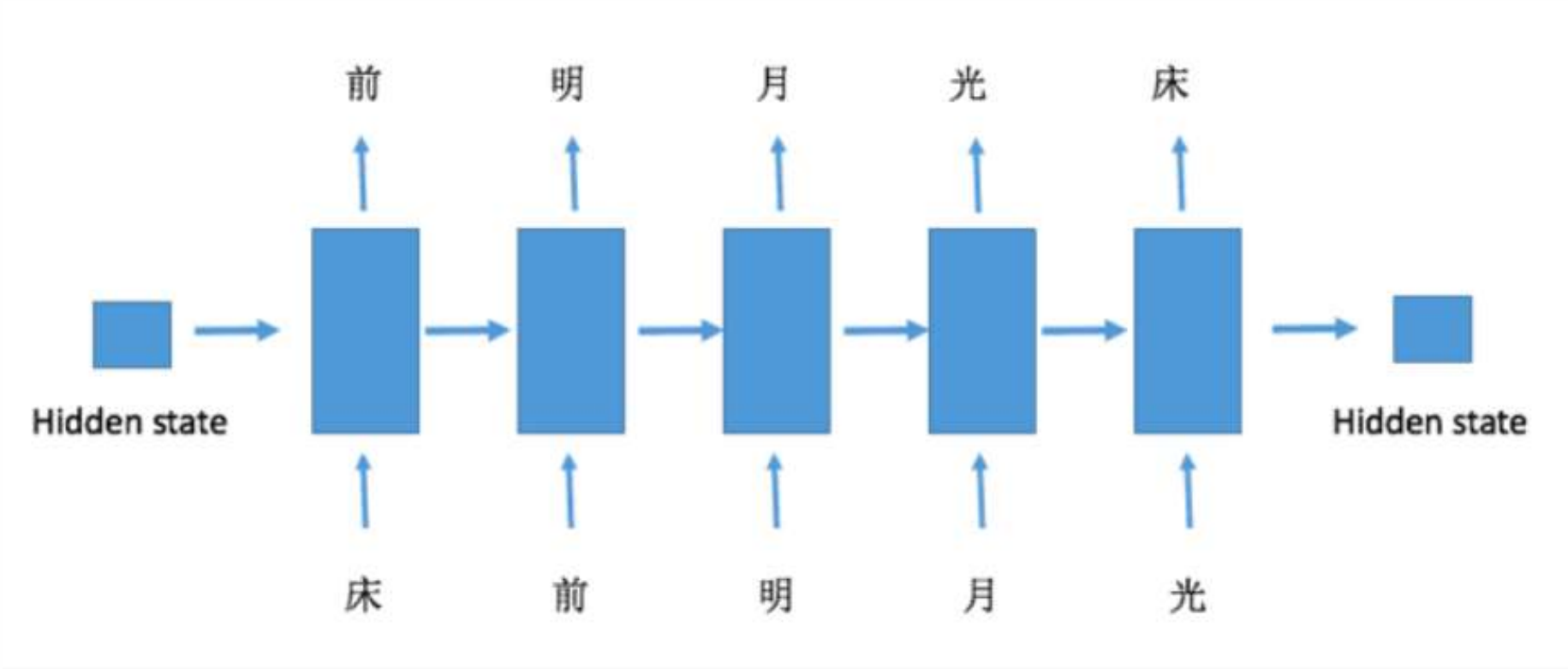


Char RNN

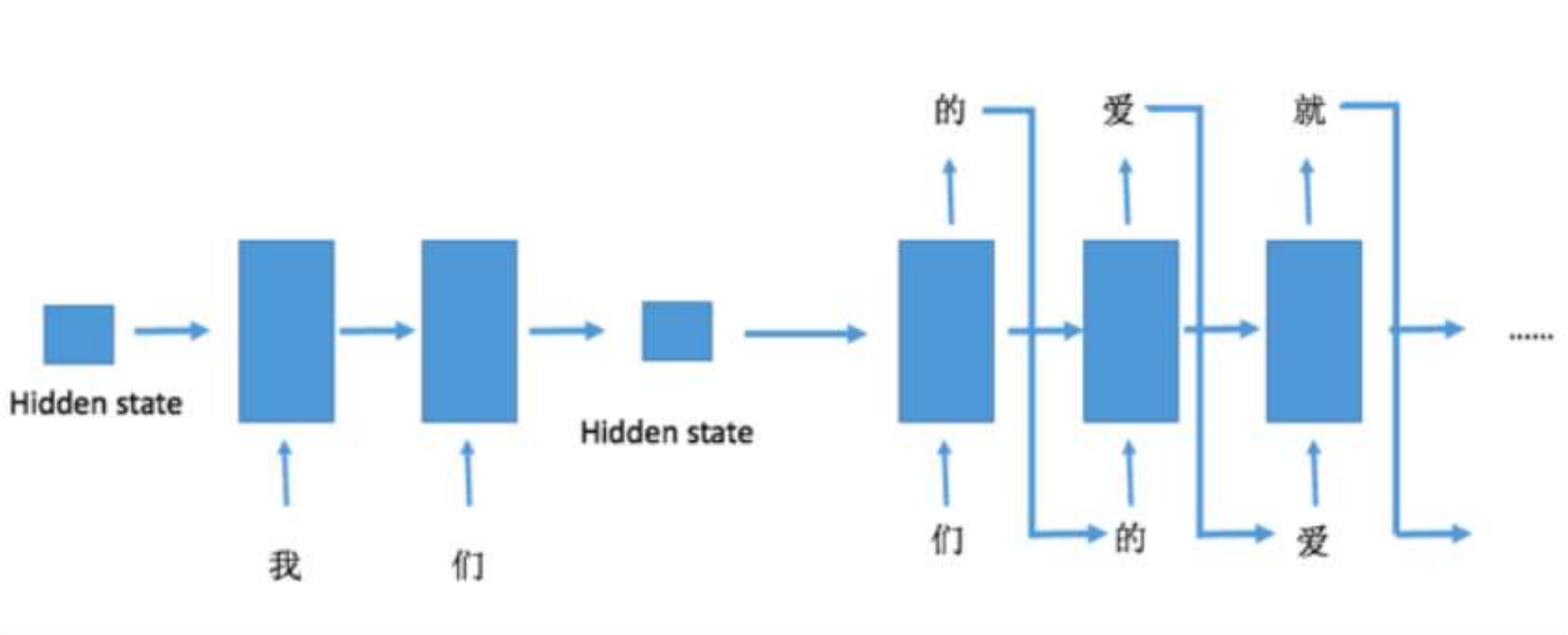
RNN的输入和输出存在着多种关系，比如1对多，多对多

在此，Char RNN是相同长度的多对多的类型，也就是输入一个序列，输出一个相同的长度的序列。

训练过程



生成文本过程



实现细节（使用PyToch训练）

1.数据预处理：

建立数字表示

设定**max_ char_num**

自定义数据集合

```
class TextData(data.Dataset):
    def __init__(self, text_path, n_step, arr_to_idx):
        self.n_step = n_step

        with open(text_path, 'r') as f:
            data = f.readlines()
        text = [v for s in data for v in s]
        num_seq = int(len(text) / n_step)
        self.num_seq = num_seq
        text = text[:num_seq * n_step]
        arr = arr_to_idx(text)
        arr = arr.reshape((num_seq, -1))
        self.arr = torch.from_numpy(arr)

    def __getitem__(self, index):
        x = self.arr[index, :]
        y = torch.zeros(x.size())
        y[:-1], y[-1] = x[1:], x[0]
        return x, y

    def __len__(self):
        return self.num_seq
```

网络定义

三层：

word embedding

RNN

线性映射

```python
class CharRNN(nn.Module):
    def __init__(self, num_classes, embed_dim, hidden_size, num_layers,
                 dropout):
        super(CharRNN, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size

        self.word_to_vec = nn.Embedding(num_classes, embed_dim)
        self.rnn = nn.GRU(embed_dim, hidden_size, num_layers, dropout)
        self.proj = nn.Linear(hidden_size, num_classes)

    def forward(self, x, hs=None):
        batch = x.size(0)
        if hs is None:
            hs = Variable(
                torch.zeros(self.num_layers, batch, self.hidden_size))
            if torch.cuda.is_available():
                hs = hs.cuda()
        word_embed = self.word_to_vec(x)  # batch x len x embed
        word_embed = word_embed.permute(1, 0, 2)  # len x batch x embed
        out, h0 = self.rnn(word_embed, hs)  # len x batch x hidden
        le, mb, hd = out.size()
        out = out.view(le * mb, hd)
        out = self.proj(out)
        out = out.view(le, mb, -1)
        out = out.permute(1, 0, 2).contiguous()  # batch x len x hidden
        return out.view(-1, out.size(2)), h0
```

值得关注：
1.向前传播的时候，指定传入的隐藏状态

2.网络输出的时候，将输出进行out.view(-1, out.size(2))这个操作

```python
class CharRNN(nn.Module):
    def __init__(self, num_classes, embed_dim, hidden_size, num_layers,
                 dropout):
        super(CharRNN, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size

        self.word_to_vec = nn.Embedding(num_classes, embed_dim)
        self.rnn = nn.GRU(embed_dim, hidden_size, num_layers, dropout)
        self.proj = nn.Linear(hidden_size, num_classes)

    def forward(self, x, hs=None):
        batch = x.size(0)
        if hs is None:
            hs = Variable(
                torch.zeros(self.num_layers, batch, self.hidden_size))
            if torch.cuda.is_available():
                hs = hs.cuda()
        #词嵌入层
        word_embed = self.word_to_vec(x)  # batch x len x embed
        word_embed = word_embed.permute(1, 0, 2)  # len x batch x embed
        #rnn
        out, h0 = self.rnn(word_embed, hs)  # len x batch x hidden
        le, mb, hd = out.size()
        out = out.view(le * mb, hd)
        out = self.proj(out)
        out = out.view(le, mb, -1)
        out = out.permute(1, 0, 2).contiguous()  # batch x len x hidden
        return out.view(-1, out.size(2)), h0
```

进行训练

训练过程非常简单，只需要把序列扔到网络中即可

值得关注：

1.将label y进行y.view(-1)

2.通过nn.utils.clip_grad_norm()
对网络进行梯度裁剪

```
for batch in dataloader:
    x, y = batch
    y = y.type(torch.LongTensor)
    mb_size = x.size(0)
    if use_gpu:
        x = x.cuda()
        y = y.cuda()
    x, y = Variable(x), Variable(y)
    out, _ = model(x)
    batch_loss = criterion(out, y.view(-1))
    # 反向传播
    optimizer.zero_grad()
    batch_loss.backward()
    nn.utils.clip_grad_norm(model.parameters(), 5)
    optimizer.step()
```

训练中：loss不断降低

```
flags.DEFINE_string('name', '9', 'name of the model')#gai1
flags.DEFINE_integer('num_seqs', 100, 'number of seqs in one batch')
flags.DEFINE_integer('num_steps', 20, 'length of one seq')
flags.DEFINE_integer('lstm_size', 128, 'size of hidden state of lstm')
flags.DEFINE_integer('num_layers', 3, 'number of lstm layers')
flags.DEFINE_boolean('use_embedding', True, 'whether to use embedding')
flags.DEFINE_integer('embedding_size', 128, 'size of embedding')
flags.DEFINE_float('learning_rate', 0.01, 'learning_rate')
flags.DEFINE_float('train_keep_prob', 0.5, 'dropout rate during training')
flags.DEFINE_string('input_file', 'data/9_New.txt', 'utf8 encoded text file')
flags.DEFINE_integer('max_steps', 5000, 'max steps to train')
flags.DEFINE_integer('save_every_n', 1000, 'save the model every n steps')
flags.DEFINE_integer('log_every_n', 10, 'log to the screen every n steps')
flags.DEFINE_integer('max_vocab', 3500, 'max char number')
```

训练时设定训练输入文件input_file、num_layers、learning_rate（0.01）、max_steps（5000）、max_vocab（3500）

```
step: 20/5000...  loss: 5.9990...  0.5242 sec/batch
step: 30/5000...  loss: 5.9902...  0.5247 sec/batch
step: 40/5000...  loss: 5.8784...  0.5717 sec/batch
step: 50/5000...  loss: 5.7664...  0.5277 sec/batch
step: 60/5000...  loss: 5.4967...  0.5007 sec/batch
step: 70/5000...  loss: 5.5118...  0.4927 sec/batch
step: 80/5000...  loss: 5.3738...  0.4877 sec/batch
step: 90/5000...  loss: 5.3961...  0.4937 sec/batch
step: 100/5000...  loss: 5.5152...  0.5327 sec/batch
step: 110/5000...  loss: 5.4150...  0.5177 sec/batch
step: 120/5000...  loss: 5.3961...  0.5127 sec/batch
step: 130/5000...  loss: 5.3045...  0.5178 sec/batch
step: 140/5000...  loss: 5.1889...  0.5747 sec/batch
step: 150/5000...  loss: 5.2709...  0.5457 sec/batch
step: 160/5000...  loss: 5.2409...  0.5727 sec/batch
step: 170/5000...  loss: 5.1857...  0.5057 sec/batch
step: 180/5000...  loss: 5.2811...  0.5767 sec/batch
step: 190/5000...  loss: 5.1285...  0.6832 sec/batch
step: 200/5000...  loss: 5.0147...  0.5347 sec/batch
step: 210/5000...  loss: 4.9516...  0.5167 sec/batch
step: 220/5000...  loss: 5.0511...  0.5407 sec/batch
step: 230/5000...  loss: 4.9979...  0.5167 sec/batch
step: 240/5000...  loss: 4.8290...  0.5217 sec/batch
step: 250/5000...  loss: 4.9187...  0.5187 sec/batch
step: 260/5000...  loss: 4.9224...  0.5041 sec/batch
step: 270/5000...  loss: 4.9963...  0.5547 sec/batch
step: 280/5000...  loss: 5.0888...  0.5207 sec/batch
step: 290/5000...  loss: 4.9441...  0.6436 sec/batch
step: 300/5000...  loss: 4.8719...  0.5667 sec/batch
```

生成文本

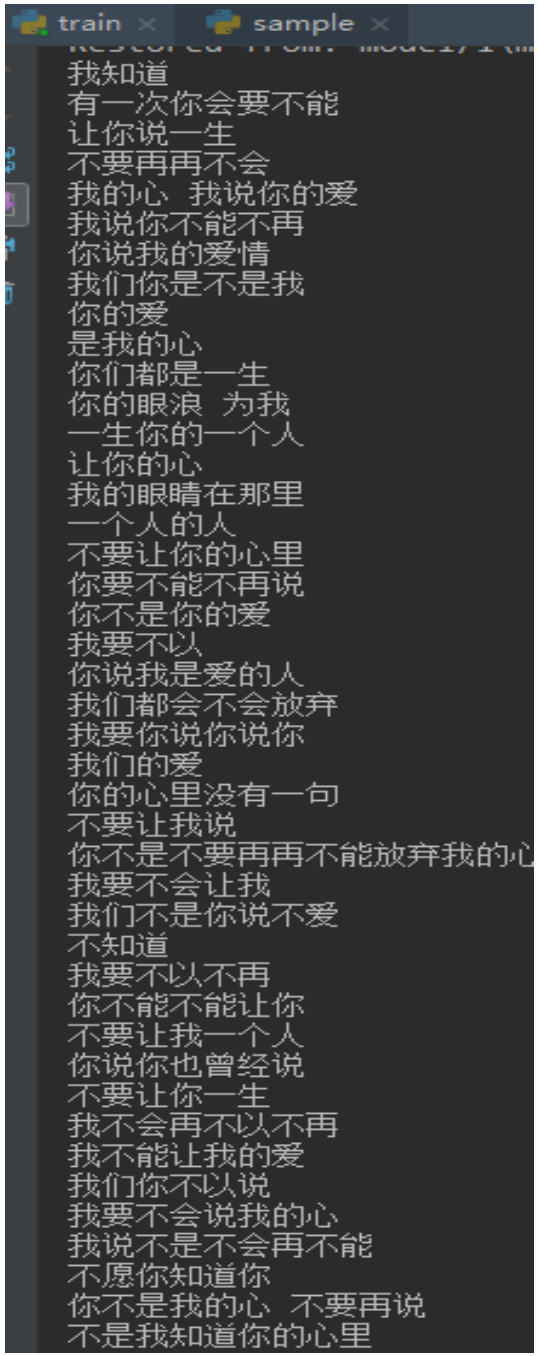
- 一句话对网络进行预热，输入隐藏状态

'start_string', '大街', 'use this string to start generating‘

- 不断循环

```
flags.DEFINE_integer('lstm_size', 128, 'size of hidden state of lstm')
flags.DEFINE_integer('num_layers', 3, 'number of lstm layers')
flags.DEFINE_boolean('use_embedding', True, 'whether to use embedding')
flags.DEFINE_integer('embedding_size', 128, 'size of embedding')
flags.DEFINE_string('converter_path', 'model/0/converter.pkl', 'model/name/converter.pkl')
flags.DEFINE_string('checkpoint_path', 'model/0', 'checkpoint path')#gai2
flags.DEFINE_string('start_string', '大街', 'use this string to start generating')
flags.DEFINE_integer('max_length', 600, 'max length to generate')
```

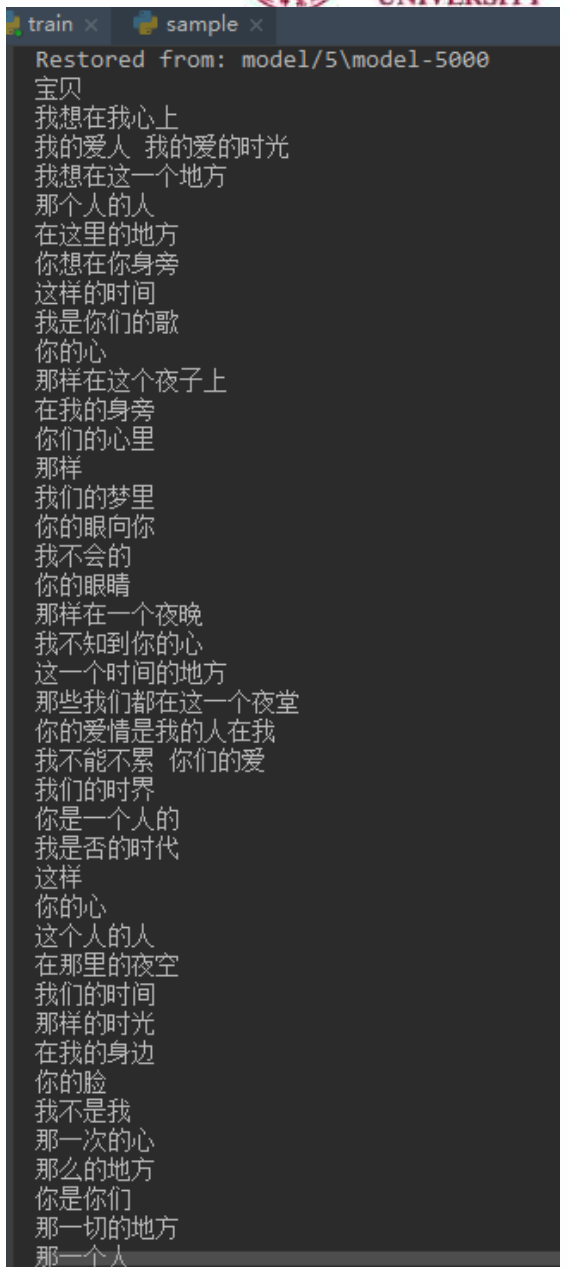
```
model.load_state_dict(torch.load(checkpoint))
model.eval()
samples = [convert(c) for c in prime]
input_txt = torch.LongTensor(samples).unsqueeze(0)
if use_gpu:
    input_txt = input_txt.cuda()
input_txt = Variable(input_txt)
_, init_state = model(input_txt) # 预热
result = samples
model_input = input_txt[:, -1].unsqueeze(1)
for i in range(text_len):
    # out是输出的字符，大小为1 x vocab
    # init_state是RNN传递的hidden state
    out, init_state = model(model_input, init_state)
    pred = pick_top_n(out.data)
    model_input = Variable(torch.LongTensor(pred)).unsqueeze(0)
    if use_gpu:
        model_input = model_input.cuda()
    result.append(pred[0])
```




```python
if __name__ == '__main__':
    start_word='宝贝'
    Category=5
    write_song(Category,start_word)
```

Result

```python
if __name__ == '__main__':
    start_word='我知道'
    Category=2
    write_song(Category,start_word)
```



宝贝

0

宝贝

你知道我的心的小小鸟

我是那个城上的小姐的小人

你是一个小小人儿

我知道的一个人

我是我的身旁的小鸟

我们的愤怒就像我想来

我要走在你的心痒

我是我

你的心点就是我的爱

你是我的小呀小苹果小苹果儿

就有我的温柔的鸟

怎么样的姑福善良

我的家

就是你的小呀小苹年

我的小呀点

Try

没了？
没了

And…

以及...
无聊时我们也用它来写了写....诗和...代码

```
何人无不见，此地自何如。
一夜山边去，江山一夜归。
山风春草色，秋水夜声深。
何事同相见，应知旧子人。
何当不相见，何处见江边。
一叶生云里，春风出竹堂。
何时有相访，不得在君心。

Process finished with exit code 0
```

```
static int page_cpus(struct flags *str)
{
    int rc;
    struct rq *do_init;
};
/*
* Core_trace_periods the time in is is that supsed,
*/
#endif
/*
* Intendifint to state anded.
*/
int print_init(struct priority *rt)
{/* Comment sighind if see task so and the sections */
    console(string, &can);
}

Process finished with exit code 0
```

華東師範大學
EAST CHINA NORMAL UNIVERSITY

Thanks

陈伟文 徐洪义 盛俊杰