基于搭配词抽取和主题模型的格律诗集句

计研 111 何巍 2011210839

hwhf6699@qq.com 13810485194

目录

1.问题描述

1.1 目标

算法的目标是生成**语句连贯**的，**主题明确**的**格律诗**。

其中，“语句连贯”主要通过搭配词抽取来保证，将在“搭配词抽取”一章详述。

“主题明确”主要通过主题模型(Topic Model)来保证，将在“主题模型”一章详述。

“格律诗”是指符合格律的平仄和押韵要求，将在“格律”一章详述。

整个集句结果的生成算法将在“集句算法”一章详述，它包括生成初始集句结果的贪心算法，和润色集句结果的寻找局部最优算法。

1.2 非目标

格律诗中的五律和七律的颔联和颈联要求符合对仗要求。如杜甫的《客至》中的标黄部分：

舍南舍北皆春水，但见群鸥日日来。

花径不曾缘客扫，蓬门今始为君开。

盘飧市远无兼味，樽酒家贫只旧醅。

肯与邻翁相对饮，隔篱呼取尽余杯

我们的算法无法保证生成的五律和七律的颔联和颈联满足对仗要求。

2.搭配词抽取

我们发现在古诗中，上下两句之间经常出现一些搭配词(collocation)。杜甫的《江畔独步寻花·其六》中有这么一句：

留连戏蝶时时舞，自在娇莺恰恰啼。

上句中加下划线的“蝶”和“莺”就经常成对出现。再比如，长孙氏的《春游曲》中有：

花中来去看舞蝶，树上长短听啼莺。

上官仪的《春日》中有：

花轻蝶乱仙人杏，叶密莺啼帝女桑。

还有很多这样的例子，直觉发现，上句中出现了“蝶”，这也就意味着下句中有很大的可能性出现“莺”。我们的任务就是验证这种直觉是否成立，以及找到古诗词中类似于<蝶->莺>这样的搭配词。我决定采用课上讲到的 t 测试。

我们把古诗分成两句一组。比如律诗共有 8 句，其中 1 和 2，3 和 4，5 和 6，7 和 8 构成了 4 组。我们的词库中共有 N=97396 组。其中上句出现“蝶”的组有 138 组，下句出现“莺”的组有 269 组。

$$P(\text{蝶}) = \frac{138}{97396}$$
$$P(\text{莺}) = \frac{269}{97396}$$

零假设是“蝶”和“莺”没有搭配关系，出现是相互独立的。那么，上句出现“蝶”，下句出现“莺”的概率是：

$$P(\text{蝶}->\text{莺}) = \frac{138}{97396} \times \frac{269}{97396} = 3.9134 \times 10^{-6}$$

而实际上，上句出现“蝶”，下句出现“莺”的组有 20 组。故 sample mean 是：

$$\bar{x} = \frac{20}{97396} = 2.0535 \times 10^{-4}$$

所以 t 测试值为：

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}} = \frac{2.0535 \times 10^{-4} - 3.9134 \times 10^{-6}}{\sqrt{\frac{2.0535 \times 10^{-4}}{97396}}} = 4.3869$$

经过查 t 检验表，可以知道 4.3869>2.576，所以我们可以有 99.5%以上的把握拒绝零假设，即认为“蝶”和“莺”是搭配词。

同样的方法，我们找到了唐诗中所有的搭配词和他们的 t-value，下图展示了 t 测试值最大的 10 个搭配词组和相应的 t 值，越大的 t 值代表了越大的成为搭配词的可能性：

有->无:17.64294816528106
无->有:16.33908277589266
千->一:13.278219547331945
金->玉:12.962900671339014
上->中:12.452490840890729
去->来:12.343815825503967
色->声:11.778563468740982
玉->金:11.647118718504208
如->似:11.336631357003121
君->我:11.279543757482815

图 1 t 值最大的 10 组搭配词

在预处理时，我们将所有 t 值不小于 1.645(对应于 95%的置信度)的搭配词组存到了一个文件 collocation.txt 中，用于下文中的集句算法。collocation.txt 的格式如下所示：

```
5 要 何必 1.672825515570504
6 覆 去 1.6831946829767763
7 黄昏 起 1.840814817856996
8 黄昏 寂寞 1.7015163786745102
9 黄昏 月 2.0488786076976178
10 黄昏 玉 1.7250913795227731
```

图 2 collocation.txt 格式示意

每一行是两个词 A 和 B，以及<A->B>对应的 t 值。

3.主题模型

我们使用机器学习时里的主题模型(Topic Model)来挖掘出唐诗中的主题。源代码中包括 plsi 和 lda 两种算法的实现。比如，我们使用 Lucene 4.5 中的 SmartChineseAnalyzer 进行分词，然后使用 plsi 算法，设定主题数为 10，可以得到各个主题下的出现概率最高的前 20 个词如下图所示：

```
Topic #1: 月 秋 夜 声 寒 云 风 孤 江 远 雨 水 山 清 思 落 空 明 梦 一
Topic #2: 边 连 城 海 马 云 汉 州 江 征 河 塞 关 出 郡 南 剑 骑 雪 秦
Topic #3: 红 玉 香 金 花 风 不 如 一 枝 似 春 轻 罗 绿 中 长 弦 露 人
Topic #4: 春 花 酒 归 醉 人 来 客 柳 别 泪 送 衣 君 看 不 去 开 愁 日
Topic #5: 三 恩 公 为 文 名 高 君 主 官 一 才 书 有 重 明 新 成 贵 知
Topic #6: 开 台 云 紫 池 回 仙 来 玉 龙 凤 天 清 日 上 绕 殿 光 阁 金
Topic #7: 不 无 何 为 有 是 时 君 身 自 事 多 我 未 得 知 亦 心 非 难
Topic #8: 山 云 鹤 不 无 到 寺 石 中 峰 人 来 寻 经 有 僧 在 松 去 一
Topic #9: 里 路 千 万 西 东 一 年 三 十 人 去 几 山 日 水 空 上 陵 望
Topic #10: 闲 竹 山 门 幽 野 静 松 卧 高 吟 深 径 鸟 石 疏 书 苔 溪 日
```

图 3 各个 topic 下出现概率最大的词

我们可以比较容易地看出各个主题下的含义。比如Topic #2 是边塞诗中常见的意象，Topic #5 中的词常出现在描述仕宦的诗中，Topic #6 中的词常用于描述亭台楼榭、琼楼玉宇。Topic #8 和 Topic #10 中的词常出现在描述山林，寻隐者之类的诗中。

有了每个词在各个主题下的概率，我们可以求出一个句子或者一首诗在各个主题上的概率分布。比如，对于边塞诗的代表作，祖咏的《望蓟门》：

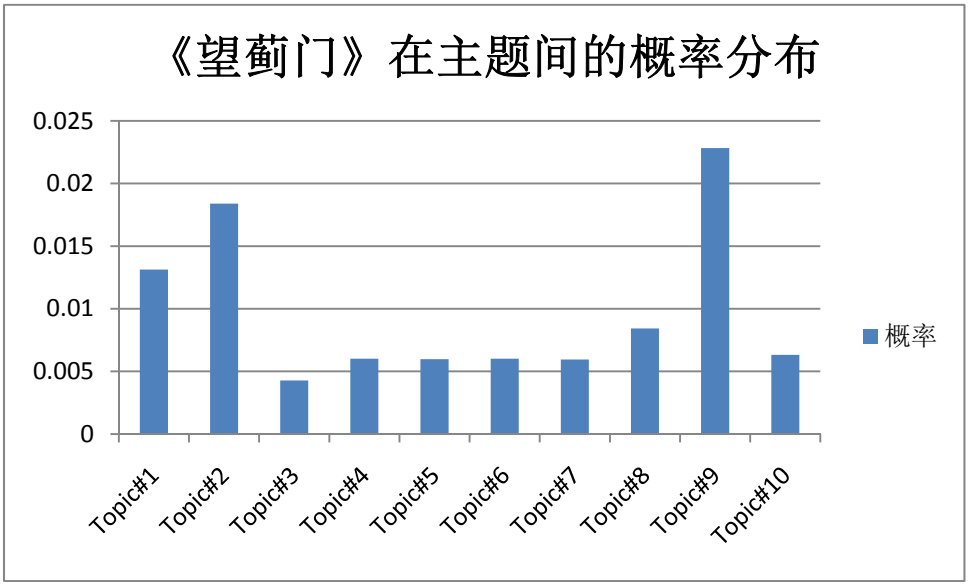
燕台一望客心惊，笳鼓喧喧汉将营。

万里寒光生积雪，三边曙色动危旌。

沙场烽火连胡月，海畔云山拥蓟城。

少小虽非投笔吏，论功还欲请长缨。

使用图 3 中训练出的主题模型，我们可以得出《望蓟门》在 10 个主题上的概率分布：



图表 1 《望蓟门》在主题间的概率分布

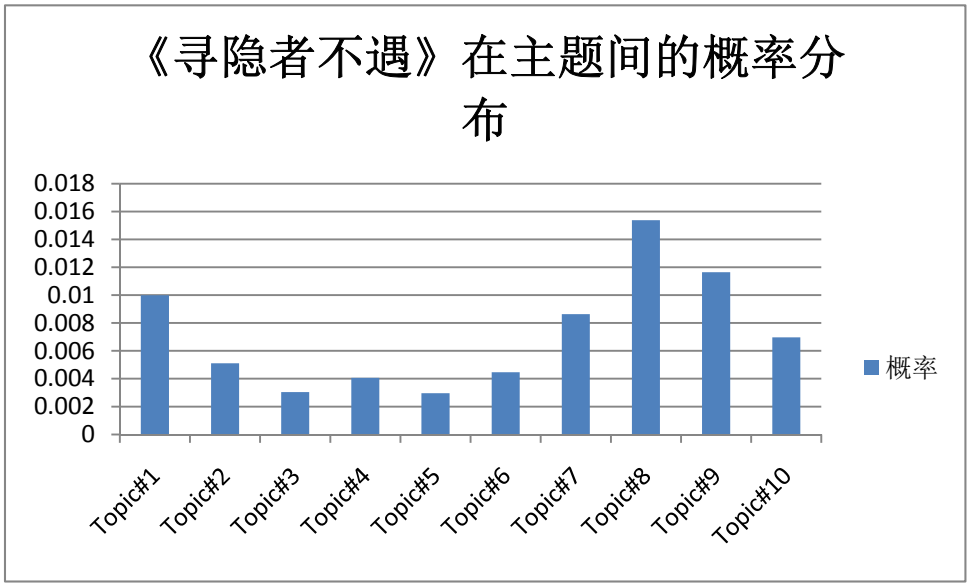
可以发现《望蓟门》该诗的概率分布在主要集中在 3 个 Topic 上，即 Topic #1, Topic #2 和 Topic #9。而其中的 Topic #2 我们之前提到过，包含了边塞诗中常见的意象。该诗在其它的主题上的概率均不大。

我们再看一首唐代寻隐诗的代表作，贾岛的《寻隐者不遇》：

松下问童子，言师采药去。

只在此山中，云深不知处。

使用图 3 中训练出的主题模型，我们可以得出《寻隐者不遇》在 10 个主题上的概率分布：



图表 2 《寻隐者不遇》在主题间的概率分布

我们同样可以发现《寻隐者不遇》的概率分布也集中在若干个主题上，其中概率最大的主题即是我们之前提到包含了寻隐诗常见意象的 Topic #8。

通过图表 1 和图表 2 的结果可以发现，诗在主题间的概率分布中，少数主题的概率之和占据了大数的空间。这隐含的含义是诗的主题应该是集中的，而非发散的。在预处理中，我们一方面将主题模型的训练结果保存在 topic.txt 文件中，便于下文的集句算法使用。另外一方面，也将主题的集中性用于指导设计集句算法。

topic.txt 格式如下图所示：

```
1 置 0.0 3.2135361962358296E-5 0.0 0.0 5.130443237969592E-5 4.5921656062689664E-274 0.0 0.0 0.0 0.0
2 要 0.0 0.0 1.338723270447136E-199 2.84038927E-315 9.700050519761096E-5 0.0 0.0011671765496352239 4.71467561
3 漏风声 0.0 0.0 0.0 0.0 0.0 0.0 7.545004743680501E-6 0.0 0.0 0.0
```

图 4 topic.txt 格式示意

每一行是一个词，和它在 N 个 Topic 下的出现概率。

4.格律

唐诗需要符合格律的要求。格律要求主要有：

a) 几乎所有的字都有平仄要求
b) 某些特定的行结尾需要入韵

比如对于“五绝仄起”，它的格律要求是：

⊙仄平平仄
平平⊙仄平（韵）

⊙平平仄仄
⊙仄仄平平（韵）

其中，“⊙”表示该处字可仄可平。“平”或“仄”表示该字必须满足平仄要求。以“(韵)”结尾的行表示该行需要入韵。

我从 http://www.52shici.com/gl.php 拿到了(五律/五绝/七律/七绝)(仄起/仄起入韵/平起/平起入韵)共 4*4=16 种格式要求，存在一个文本文件 pattern.txt 中。pattern.txt 的格式如下图所示：

```
#五律仄起
⊙仄平平仄
 平平仄仄平x
⊙平平仄仄
⊙仄仄平平x
⊙仄平平仄
 平平⊙仄平x
⊙平平仄仄
⊙仄仄平平x

#五律仄起入韵
⊙仄仄平平x
 平平⊙仄平x
⊙平平仄仄
⊙仄仄平平x
⊙仄平平仄
 平平⊙仄平x
⊙平平仄仄
⊙仄仄平平x
```

图 5 pattern.txt 格式示意

关于如何检测一个字是平是仄，需要注意到字的现代读音和古音是不同的。我们以古音为准，用到了《平水韵》。平水韵依据唐人用韵情况，把汉字划分成 106 个韵部（其书今佚）。每个韵部包含若干字，作律绝诗用韵，其韵脚的字必须出自同一韵部。其中若干个韵部中的字属于平声，其余韵部的字属于仄声。我从网上找了一份平水韵表，保存在了一个文件文件 pingshuiyun.txt 中。它的格式如下图所示：

```
一东，东同铜桐筒童僮疃中衷忠虫终戎崇嵩弓躬宫融雄熊穹穷冯风枫丰充隆空公功工攻蒙笼聋珑洪红鸿虹丛翁聪通蓬烘潼胧砻峒鑫
二冬，冬农宗钟龙舂松冲容蓉庸封胸雍浓重从逢缝踪茸峰锋烽蛩慵恭供淙侬松凶塘镛佣溶邛共憧喁邕壅纵龚栊脓淞匈汹禹蚣榕彤
三江，江扛窗邦缸降双庞逢腔撞幢桩淙豇
```

图 6 pingshuiyun.txt 格式示意

5.集句算法

5.1 目标函数

在前文中提到的，我们的目标是生成**语句连贯**，**主题明确**的**格律诗**。其中的格律诗的要求是硬约束，不符合格律要求的集句结果不会出现在结果集中。而对于“语句连贯”和“主题明确”，我们通过目标函数来约束。

为此我们定义两个 collocationScore 和 topicConcentrationScore:

1) collocationScore 衡量格律诗的语句连贯程度。
2) topicConcentrationScore 衡量格律诗的主题集中程度。

计算 collocationScore 的公式如下：

$$collocationScore(poem)=\frac{\sum_{i=0}^{\frac{poem.row}{2}-1}\ \underset{term1\in poem.lines[2i+1],term2\in poem.lines[2i+2]}{Max} t-value(term1->term2)}{\frac{poem.row}{2}}$$

公式的意义如下：如对于绝句，可分为两句一组。对于第 1 句和第 2 句，寻找第一句包含的一个词 t1，和第二句包含的一个词 t2，使得 t1 和 t2 搭配的 t-value 值最大。同样地，我们也可以找到第 3 句中的一个词 t3 和第四句中的一个词 t4，使得 t3 和 t4 搭配的的 t-value 值最大。那么，t-value(t1->t2)和 t-value(t3-t4)的平均值即为 collocationScore。

也就是说，在一个诗中，collocationScore 越大，表明每两句一组中出现了越常见的搭配词。

计算 topicConcentrationScore 的公式如下，设 prob 为该诗在所有主题间的概率分布，且按照概率值的大小由高到低排序：

$$topicConcentrationScore(poem)=\frac{\sum_{i=1}^{K} prob[i]}{K\sum_{i=1}^{nTopic} prob[i]}$$

这个公式的意义非常明确：topicConcentrationScore 越大，表示该诗的主题间的概率分布越集中在 Top K 个主题上。当 prob 全部集中 Top K 个主题上时，topicConcentrationScore 的值最大，为 1/K。作为对照，当 prob 为均匀分布，即诗在主题间没有表现出集中性时，topicConcentrationScore 值最小，为 1/nTopic。

集句结果的得分是 collocationScore 和 topicConcentrationScore 的加权求和，即：

$$Score(poem) = \lambda * topicConcentrationScore + collocationScore$$

λ 调节 collocationScore 和 topicConcentrationScore 之间的相对重要程度。下面的算法即为找到一个 Score 值最大的集句结果。

5.2 生成初始集句结果的贪心算法

生成初始集句结果我们采用贪心算法，该算法依次产生集句中的每一句，其中每一句均需要符合格律要求，且按下面的要求产生：

1) 第 1 句随机产生。
2) 产生偶数句，即第 2、4(、6、8)句时，我们需要保证它和前一句，即第 1、3(、5、7)句有 t-value 最高的搭配对。
3) 产生奇数句，即第 3(、5、7)句时，需要保证它和前面的所有句组成的整体在主题间的概率分布的 topicConcentrationScore 最高。

伪代码如下：

获取初始集句结果(格律要求 pattern)

从满足 pattern 要求的候选集中随机选取第一句

for i <- 2 to pattern.行数要求

S<-满足 pattern 要求的候选集

if i 是奇数

从 S 中选取能最大化 topic concentration score 的 line

poem.lines[i]=line

else

从 S 中选取能最大化 $\underset{term1 \in poem.lines[i-1], term2 \in line}{Max} t-value(term1->term2)$ 的 line

poem.lines[i]=line

5.3 润色集句结果的寻找局部最优算法

产生了初始结果集后，我们需要对其进行“润色”，即进行调优，以最大化 Score 值。调优最多进行固定的轮数，在每一轮中，对于诗的每一句，在固定其它句的前提下，我们试图将其替换成一个新的句子，新句同样需要符合格律要求，同时能够最大化 Score 值。如果在某一轮调整中，没有进行新的变动，则说明我们已经找到了一个局部最优解。伪代码如下：

润色集句结果(Pattern pattern, Poem poem)

```
for iter<-1 to 固定轮数
    for i<- 1 to poem.行数
        S<-满足 pattern 要求的候选集
        计算 S 中的各候选和 poem.lines[1…i-1,i+1…row]组成的新诗的 Score 值
        if 存在候选能够提高 Score 值
            从 S 中选择能最大化 Score 的 line
            poem.lines[i] = line
    if 该轮中各句均没有替换发生
        break
```

6.结果

6.1 语料库

我从新浪爱问(http://ishare.iask.sina.com.cn/)上下载了一个 txt 格式的全唐诗，但该 txt 中的格式较乱，且包含了大量的非格律诗。为此，我写了一个程序，将其转换成了标准格式，且去掉了原始语料库中的非格律诗。整理后的语料库在一个大小为 3.6M 的 poem.txt 文件中，其格式如下：

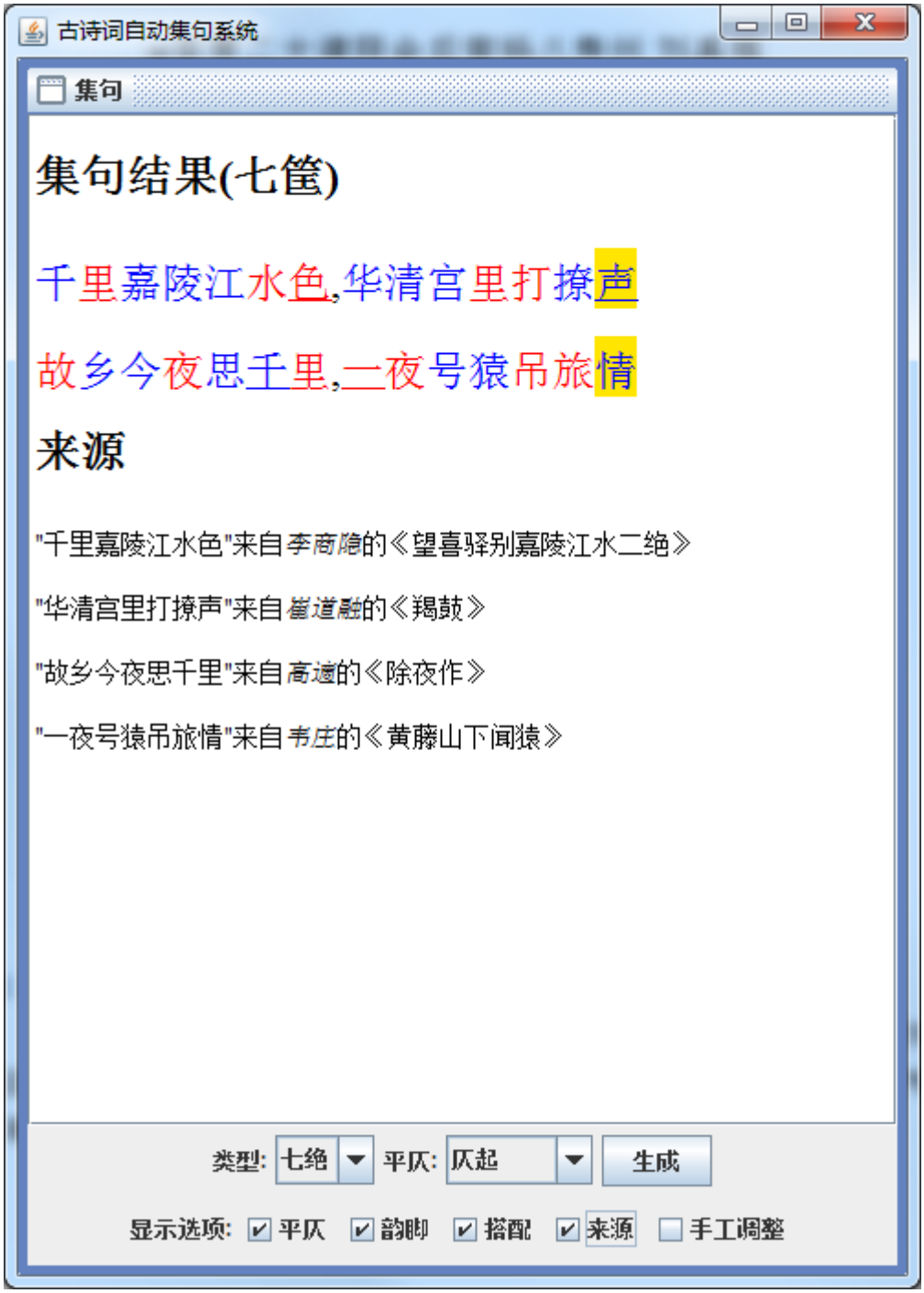
```
#李贾二大谏拜命后寄杨八寿州 刘禹锡
谏省新登二直臣
万方惊喜捧丝纶
则知天子明如日
肯放淮南高卧人
%七绝仄起入韵 十一真 0 0

#美温尚书镇定兴元以诗寄贺 刘禹锡
旌旗入境犬无声
戮尽鲸鲵汉水清
从此世人开耳目
始知名将出书生
%七绝平起入韵 七筐 0 0

#酬瑞州吴大夫夜泊湘川见寄一绝 刘禹锡
夜泊湘川逐客心
月明猿苦血沾襟
湘妃旧竹痕犹浅
从此因君染更深
%七绝仄起入韵 十二侵 0 0
```

图 7 poem.txt 格式示意

其中#打头的行，包含了该行的题目和作者。%打头的行包含了该诗的格律信息，即使用的平仄、韵部，以及错平仄和错韵的个数。

6.2 测试结果

我写了一个图形化界面的 poem composer，它可以允许用户根据指定的格律生成集句，并且标识出各字的平仄、韵脚、搭配词的选取，以及各句的来源，并且允许用户根据机器推荐手工调整集句结果。如下图所示：



、

图 8 poem composer 截图

该工具的使用方法请见《用户手册》，在此不详述。我们在此分析上面的集句结果是如何生成的，以及在生成过程中 Score, CollocationScore 和 Topic ConcentrationScore 是如何变化的。

我们选用的参数是 $\lambda = 30$，即

$Score(poem) = 30 * topicConcentrationScore + collocationScore$， $K = 2$，即我们优化主题间概率分布，使其尽可能集中地分布在概率最高的两个主题上。

获取的初始集句结果是：

作活每常嫌费力，惟天为大阐洪名。

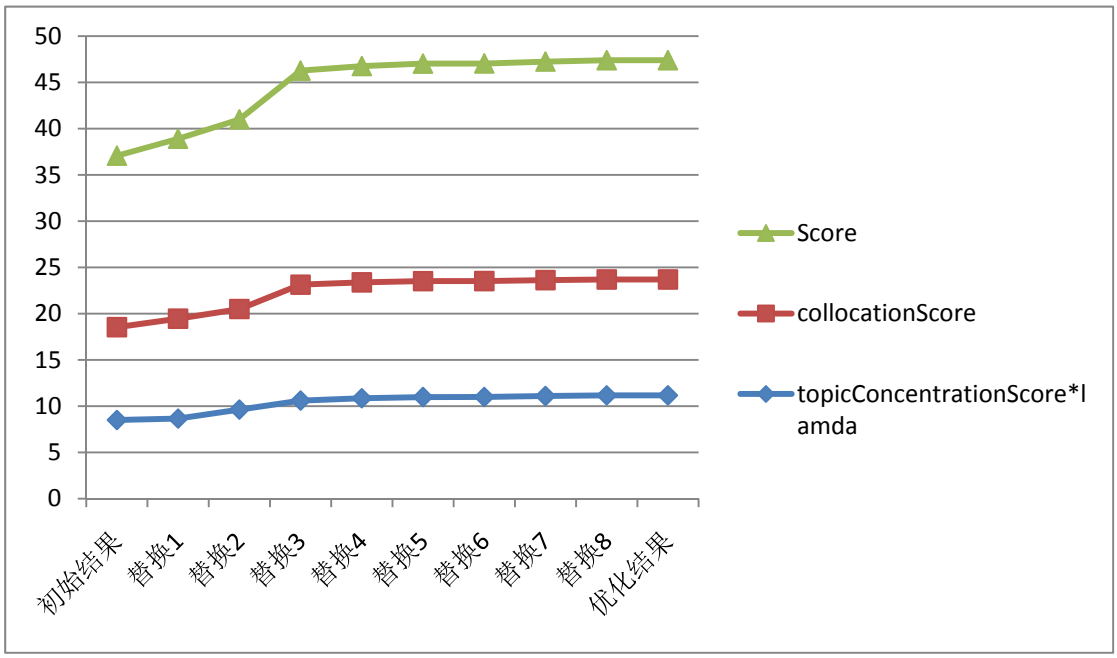
京华不啻三千里，树里南湖一片明。

它的 Score 是 18.536854807148615，其中 topicConcentrationScore 是 0.2832742832631988，collocationScore 是 10.038626309252649。我们可以看出，这样的一首诗，虽然符合平仄要求，但是实际上主题并不明确。调优过程会对它进行优化，如下表所示：

轮数	动作	topicConcentrationScore	collocationScore	Score
0	作活每常嫌费力，惟天为大阐洪名。 京华不啻三千里，树里南湖一片明。	0.2832742832631988	10.038626309252649	18.536854807148615
1	作活每常嫌费力->辇路已平栽药地	0.28836178444715727	10.801428397572945	19.452281930987667
1	惟天为大阐洪名->犹陪落日泛秋声	0.32102356927858944	10.868893488355504	20.499600566713188
2	辇路已平栽药地->千里嘉陵江水色	0.3534754584007133	12.528391508036462	23.132655260057786
2	犹陪落日泛秋声->冲湾泼岸夜波声	0.36170227108340536	12.528391508036462	23.379459640538862
2	京华不啻三千里->故乡今夜思千里	0.36614832238172257	12.528391508036462	23.512841179488814
2	树里南湖一片明->帆挂秋风一信程	0.36639258248675494	12.528391508036462	23.520168982639106
3	冲湾泼岸夜波声->华清宫里打撩声	0.36976390060428305	12.528391508036462	23.621308526164952
3	帆挂秋风一信程->一夜号猿吊旅情	0.37229741147101864	12.528391508036462	23.697313852167026
4	千里嘉陵江水色，华清宫里打撩声。 故乡今夜思千里，一夜号猿吊旅情。	0.37229741147101864	12.528391508036462	23.697313852167026

图表 3 优化过程

在优化过程中，topicConcentrationScore、collocationScore 和 Score 均会增加，其变化如下图所示 ：



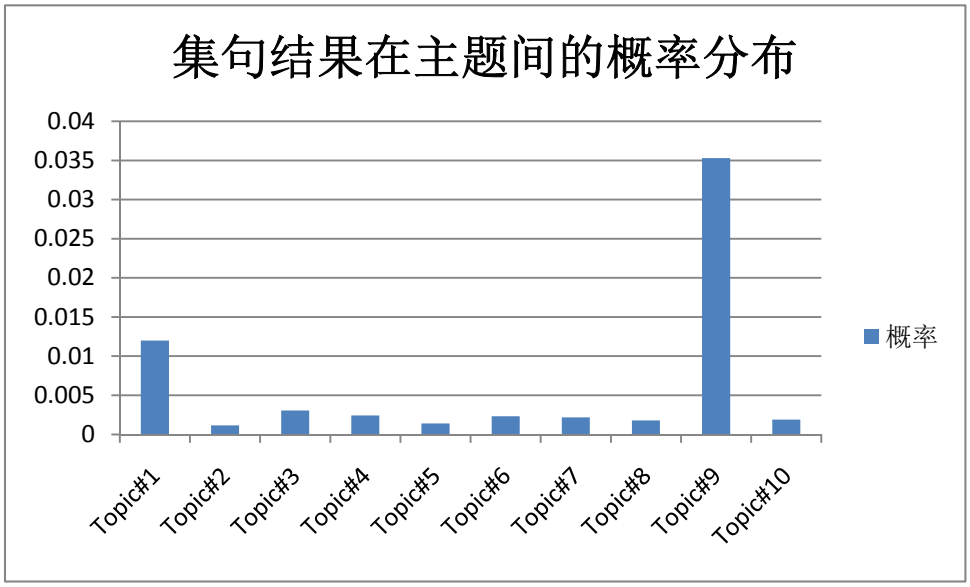
图表 4 优化过程中各评估值的变化

我们可以从上图中看出，在上图中，每次替换操作带来的值增加逐渐变小，三个评估值逐渐趋于稳定。在第 4 轮的时候，没有新的替换发生，优化结束。因此，我们找到一个局部最优解。新的集句结果是：

千里嘉陵江水色，华清宫里打撩声。

故乡今夜思千里，一夜号猿吊旅情。

可以看出，润色后的集句结果，无论是在语句连贯、还是在主题明确上，都明显优于润色前的结果。其中，第 1 句中的“色”和第 2 句的“声”，第 3 句中的“千”和第 4 句中的“一”均是较常见的搭配词组。而整首诗在各个主题间的概率分布如下图所示：



图表 5 集句结果在主题间的概率分布

可以看出，集句结果主题的概率分布主要集中在了 Topic #1 和 Topic #9 上，这一点符合我们设置 K=2 的预期。