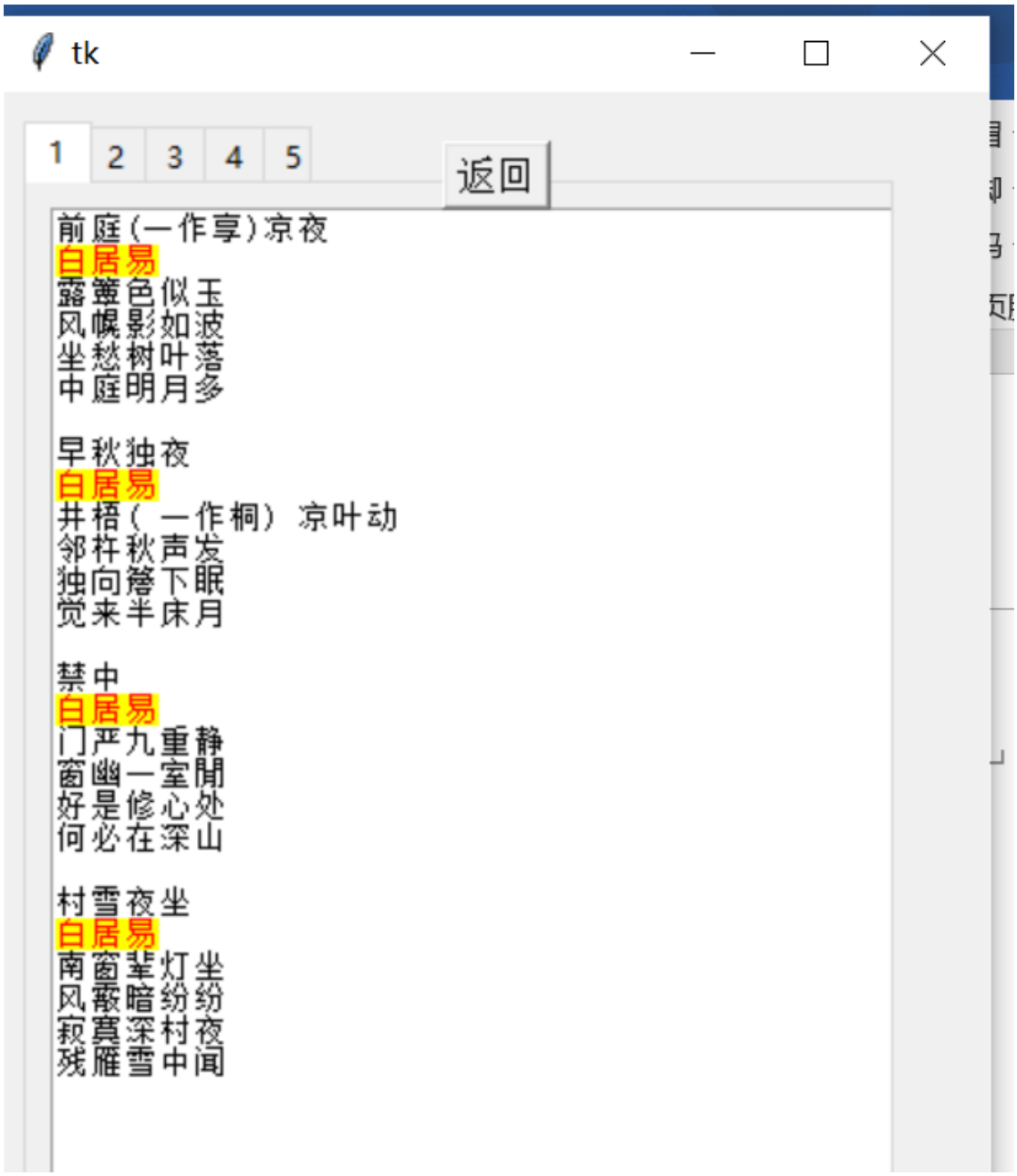
Report

1700015878 张静远

一、实现的功能

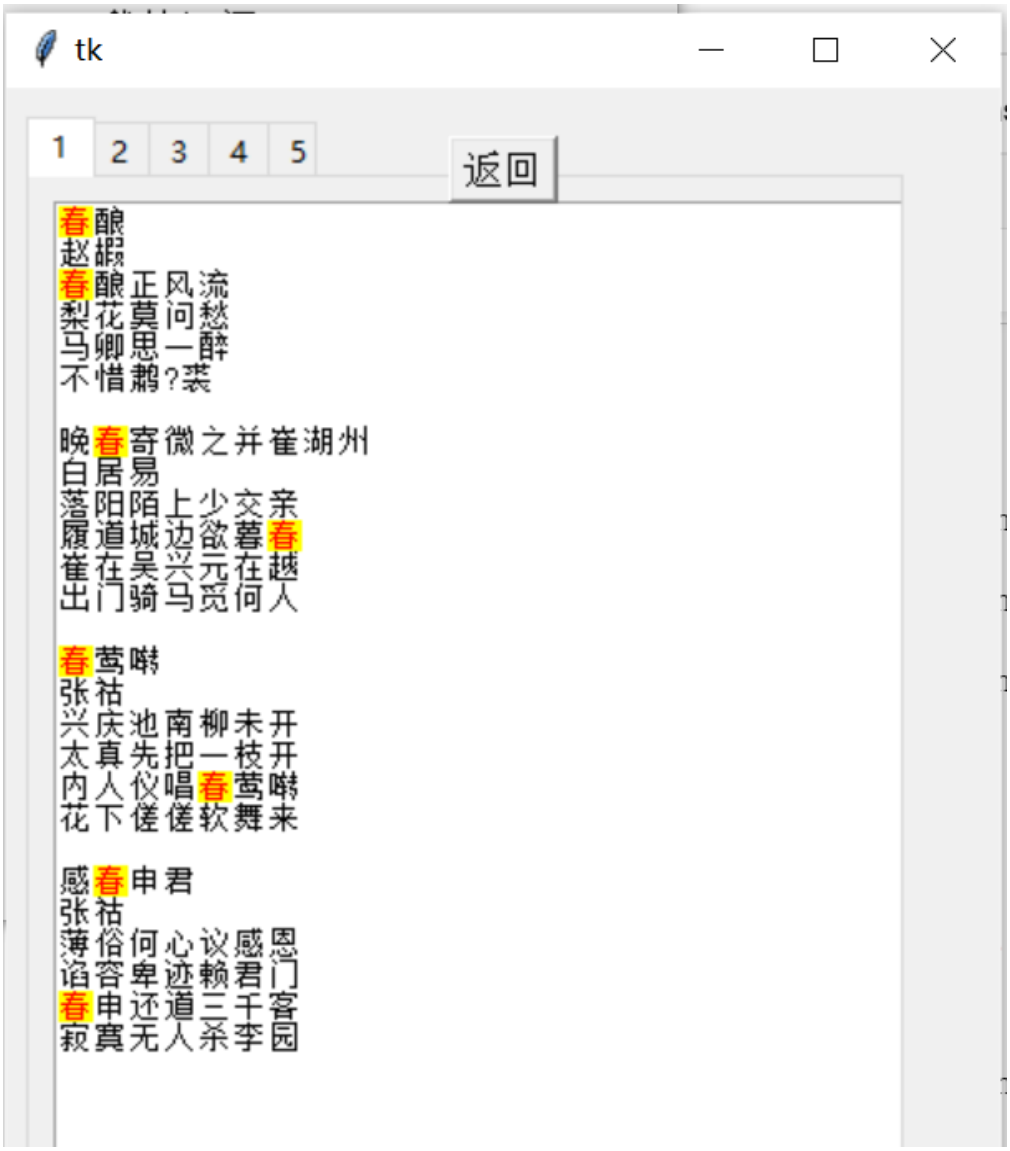
1.根据作者订阅

输入合法的作者名字，返回作品库中该作者所有的作品（只展示五页）



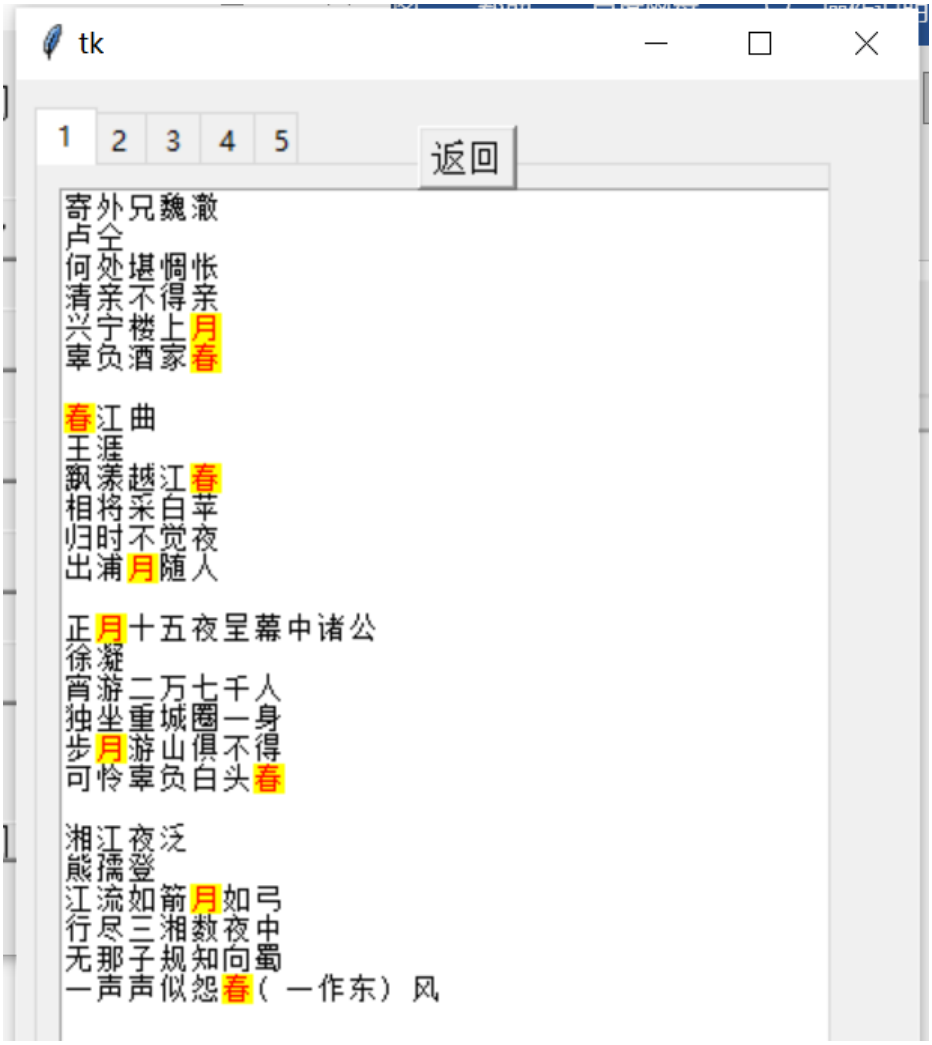
2.根据输入的关键词订阅，支持多关键词（1-3 个）

1）如果只输入一个关键词



先返回所有包含该关键词的诗歌，如果返回的诗歌数目少于MAX_POEM_SHOWN（例如 20），对关键词进行近义词挖掘。例如，输入关键词“明月”，除了返回包含“明月”的诗外，还要返回包含明月的近义词的诗（按置信度将返回的诗歌排序）。

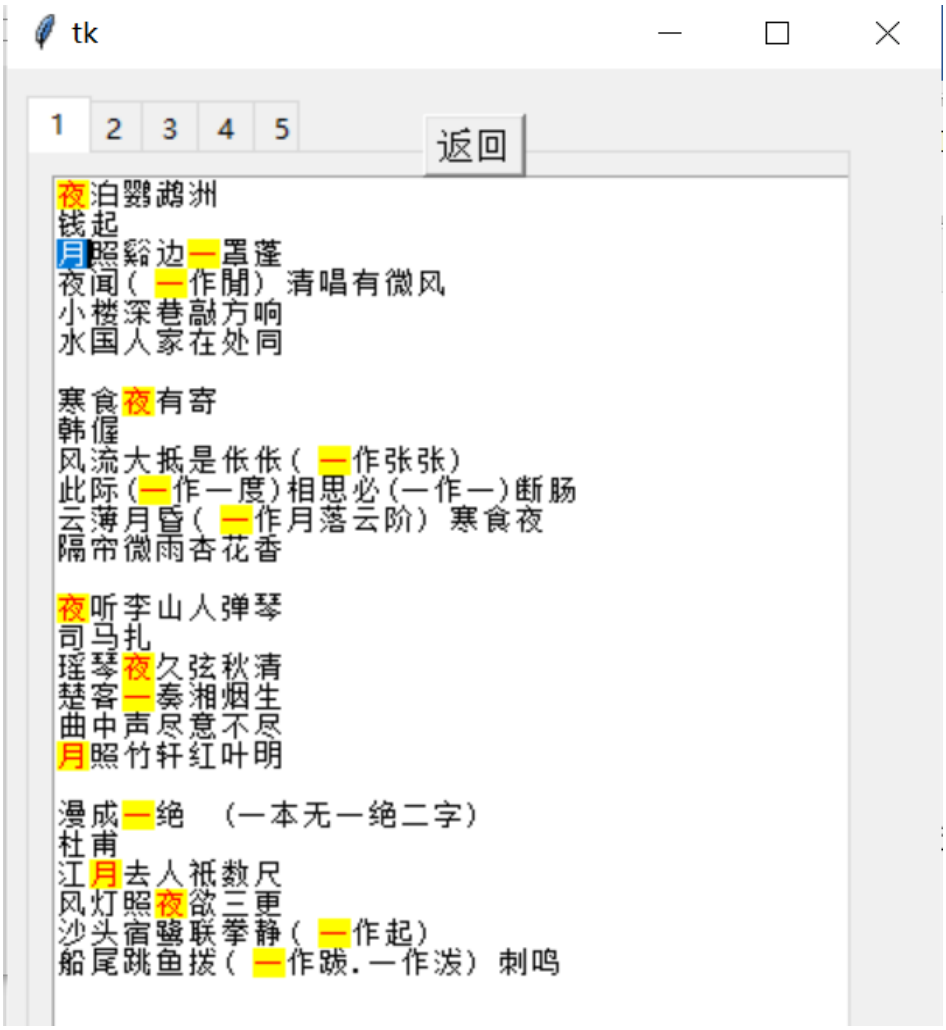
2）如果输入两个关键词，诗歌分为以下几种，进行一定加权后按权重返回



✓ 同时包含 2 或 1 个关键词的诗歌

✓ 两个关键词其中 1 或 2 个被近义词替代的诗歌

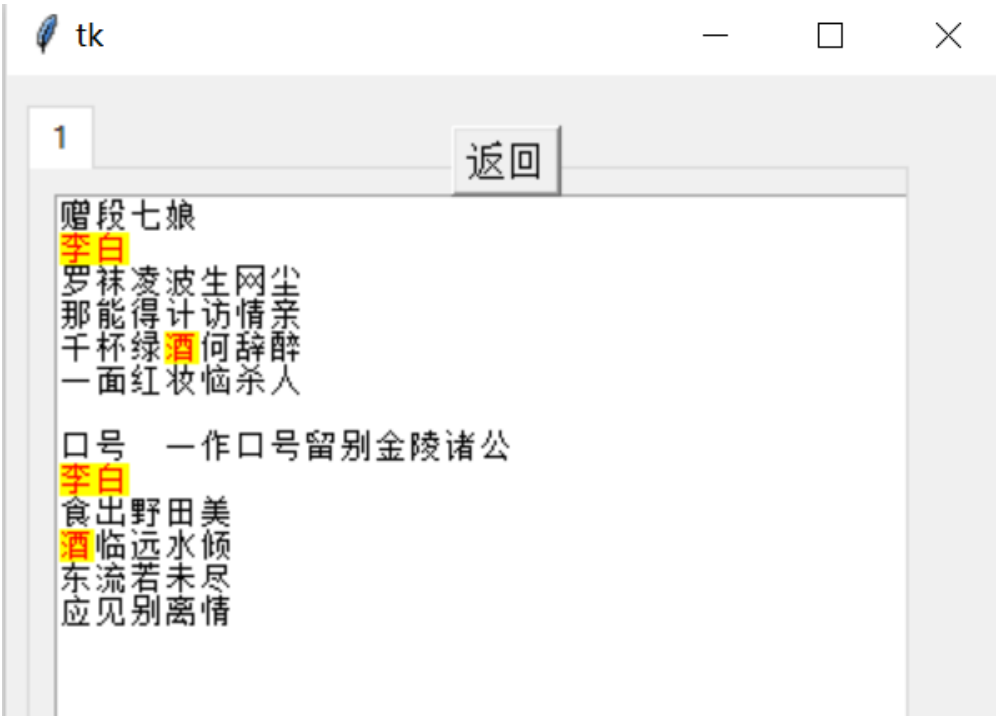
3）如果输入三个关键词，诗歌分为以下几种，进行一定加权后按权重返回



✓ 同时包含 3 或 2 或 1 个关键词的诗歌

✓ 1 个或 2 个或 3 个关键词被近义词替代

3. 同时订阅作者和关键词，返回作者的诗集中符合关键词查询结果的所有诗歌



4.订阅时，将返回结果的诗人名或关键词/关键词的近义词高亮。

5. 订阅者在查看内容时可以翻页（点击上方页码）

二、文档 TF-IDF 特征提取

1.关键词本身 TF_IDF 特征提取

构建一个 N*K 的矩阵，N 表示分词的个数，K 表示所有诗歌总数

1）$TF_{d,a}$:分词 a 在文档 d 中出现的频率

2）$IDF_a = \log(\text{K/分词 a 在所有文档中出现的频率})$

3）$weight_{d,a} = TF_{d,a} * IDF_a$，表示文档 d 在关键词 a 查询时的优先权重

当搜索包含某个关键词的文档时，按照 weight 排序返回相应的文档

三、近义词挖掘

1.算法思路

1）对词频超过 10（所有诗中的总频度）的词进行近义词挖掘。推荐通过词在诗

歌中的上下文分布向量（TF-IDF）的相似度来得到词汇间的相似关系。（注意：近义词挖掘只限制在词频大于 10 的词汇之间，但计算特征向量时要考虑所有词频大于 1 的词汇）

2）由于诗歌本身篇幅较短，认为上下文的范围是整篇诗歌

3）上下文矩阵的构建

- ✓ 利用上下文（如前后相邻）的词语，构造一个上下文矩阵。此时 tf 和 idf 的含义略有变化，tf[a,b] 表示词 b 出现在词 a 的上下文的频率，idf[b] 中的 df 的含义从“词 b 出现在多少个文档中”变成“词 b 出现在多少个词的上下文中”。
- ✓ 可以认为同一篇出现的词权重相等，也可以根据在上下文中的距离，进行适当的加权，比如认为相关度和距离的开方成反比，计算 tf 时乘以 1/sqrt(dist(a,b)) 作为距离权重。

4）计算衡量分词之间的相关度：用上下文矩阵中两个分词向量的余弦来表示，进行排序

2.评价方案

评价的基准是 Word2vec 的训练结果，比较取相同参数时本模型训练结果和 word2vec 训练结果的区别。具体衡量指标有：

前 5、10、30、50、100 个近义词中相同的比例取均值，按一定比例求和（0.3，0.25，0.2，0.15，0.1），作为评分。

3.近义词挖掘方案比较

1）上下文的范围

距离没有影响，取等权

N	5	10	30	50	100	总分
3	0.001389	0.0026325	0.005850	0.009872	0.01814	0.0055396
5	0.002998	0.005265	0.009213	0.013162	0.02179	0.0082115
10	0.018135	0.024789	0.038391	0.048555	0.07196	0.0337952
15	0.034003	0.045265	0.072468	0.092431	0.13989	0.0638644

可以看出 N=15 时效果较好，在一定范围内，上下文范围增加，总的预测精度增加。

2）根据距离加权

控制上下文范围为 N=10

- ✓ 没有加权：同一段上下文的分词等权
- ✓ 用$\frac{1}{distantce(i,j)}$加权
- ✓ 用$\frac{1}{sqrt(distantce(i,j))}$加权

N	5	10	30	50	100	总分
等权	0.018135	0.024789	0.038391	0.048555	0.07196	0.0337952
1/distance	0.015503	0.021572	0.035904	0.046801	0.06873	0.03111785
1/sqrt(dist)	0.017038	0.02340	0.037806	0.048994	0.07217	0.0330887

距离加权影响不大，等权和$\frac{1}{sqrt(distantce(i,j))}$加权结果类似，略优于$\frac{1}{distantce(i,j)}$加权。

3）分词长度的权重

当分词长度不同时，给权重乘以一个惩罚系数.

控制上下文范围 N=10，距离加权方式为等权

Punish_coef	5	10	30	50	100	总分
0.8	0.018135	0.024789	0.038391	0.048555	0.07196	0.033795
0.6	0.017623	0.02457	0.038829	0.049725	0.074369	0.03409085

可以看出给长度不一的分词给予更高的惩罚有助于提高预测精度。

四、返回结果加权方案

1.如果只输入一个关键词

先返回所有包含该关键词的诗歌，如果返回的诗歌数目少于 MAX_POEM_SHOWN（例如 20），返回关键词的近义词诗歌。

2.如果输入两个关键词，诗歌分为以下几种，进行一定加权后按权重排序从高到低返回

1）同时含有两个关键词的文档如何排序：需要重新计算权重。我们认为第一个关键词的权重高于第二个词，因此$weight_{intersection} = \alpha * weight_1 + (2-\alpha) * weight_2$。此处$\alpha$是一个需要优化的参数。

2）两个关键词其中 1 被近义词替代的诗歌(type1)，两个关键词都被近义词替代的诗歌(type2)，只含有一个关键词的诗歌(type3)。

- ✓ 当一个关键字被替代后，将该文档的权重乘以关键词和其近义词的相关系数，然后对两个文档进行 1）中的加权计算

3.如果输入三个关键词，诗歌分为以下几种，进行一定加权后按权重从高到底返

回

1）同时含有三个关键词的文档如何排序：需要重新计算权重。我们认为第一个关键词的权重高于第二个词高于第三个词，因此$weight_{intersection} = \alpha * weight_1 + \beta * weight_2 + (3 - \alpha - \beta) * weight_3$。此处$\alpha$和$\beta$是需要优化的参数。

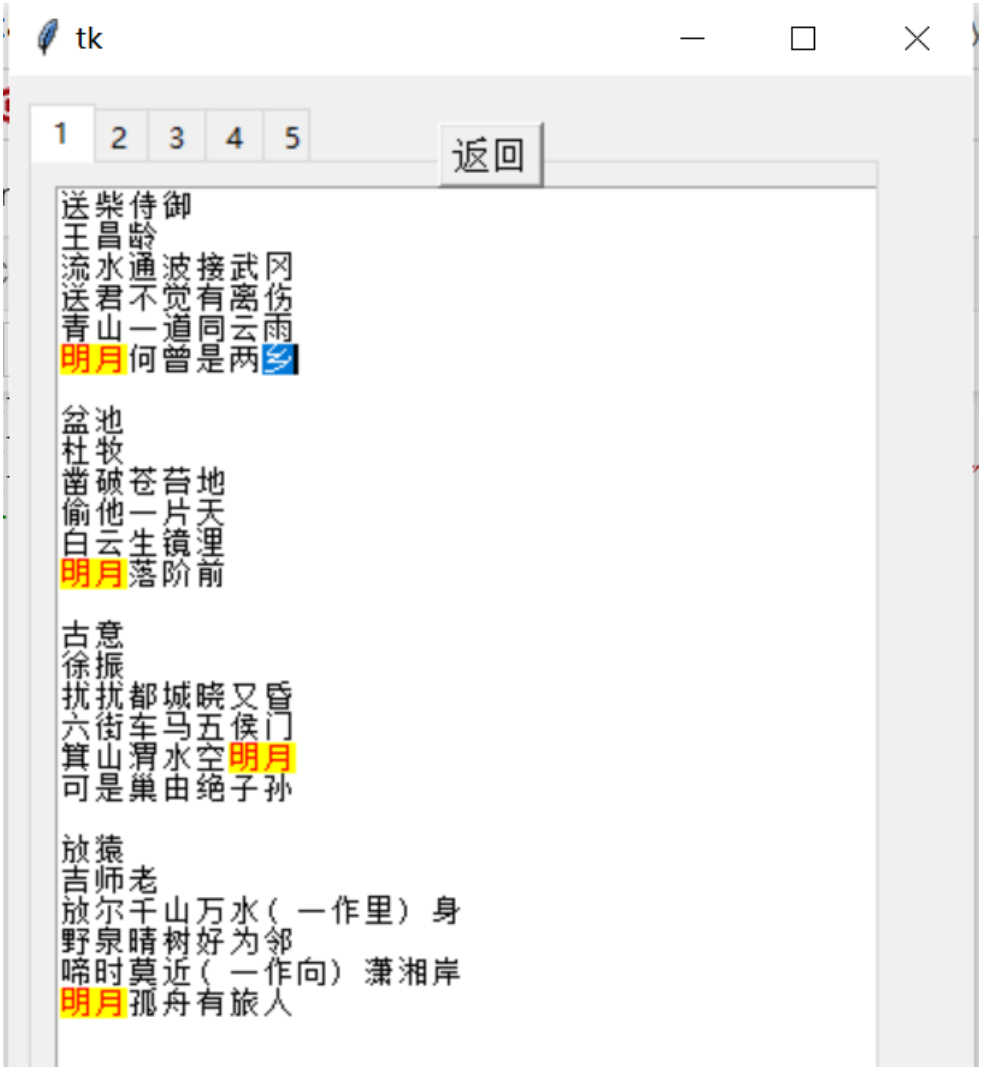
2）三个关键词其中 1 被近义词替代的诗歌(type1)，两个关键词被近义词替代的诗歌(type2)，只含有一个关键词的诗歌(type3)。

- ✓ 当一个关键字被替代后，将该文档的权重乘以关键词和其近义词的相关系数，然后对两个文档进行 1）中的加权计算

五、关键词扩展前后结果的对比分析

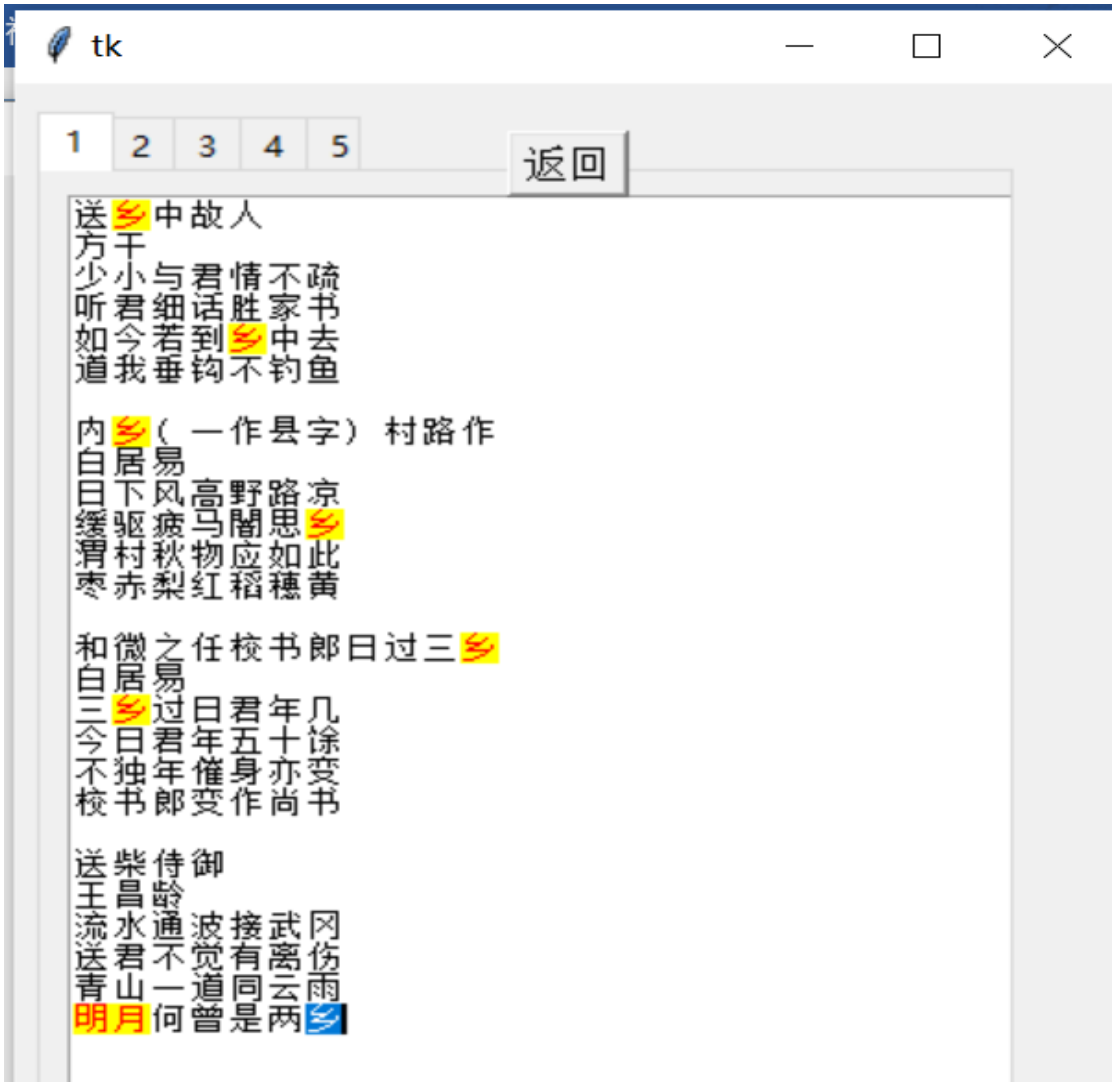
1.扩展前

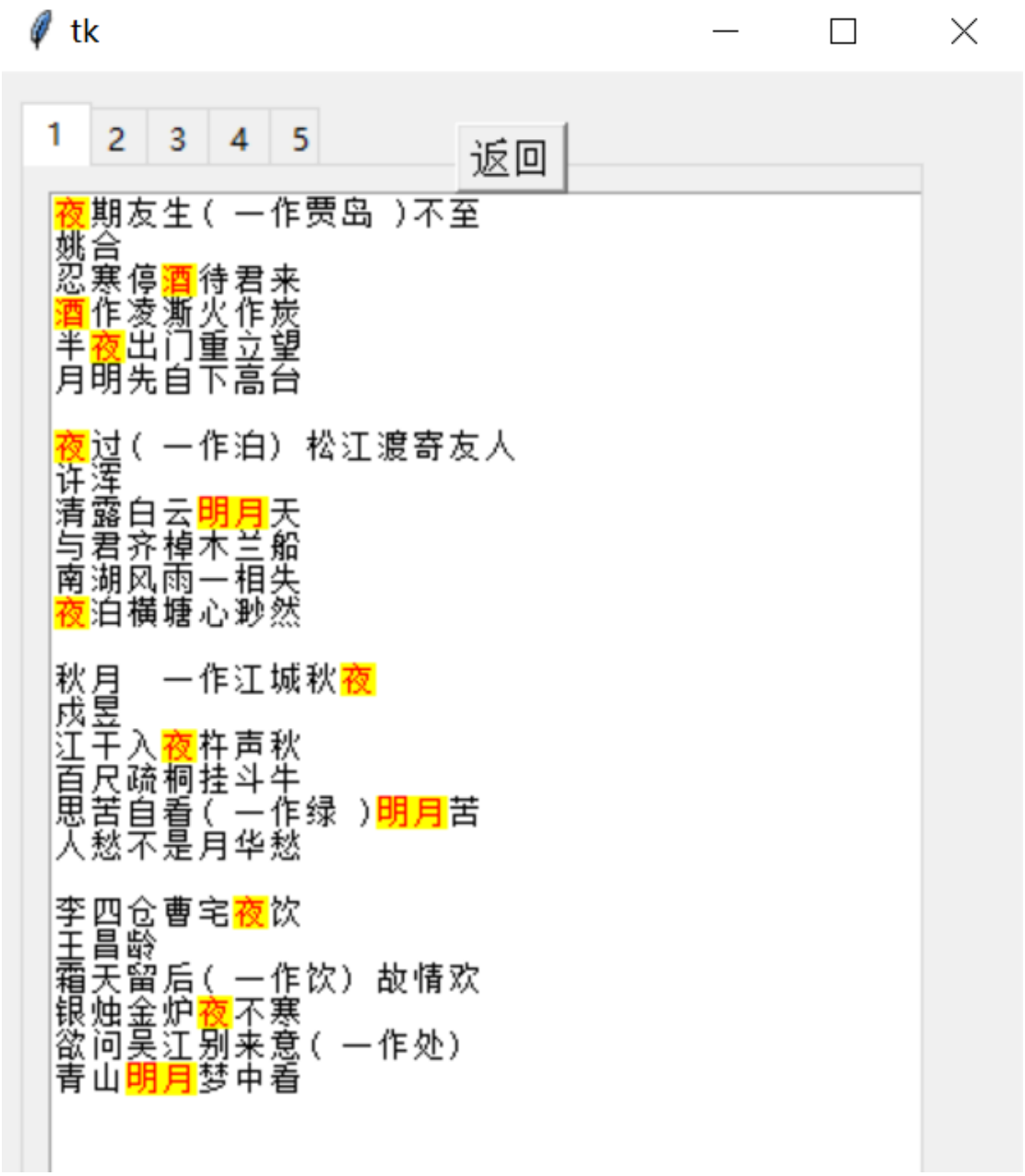
同时包含“明月”和“乡”的诗歌只有一首，扩展前除了返回同时包含两个关键词的诗歌，接下来返回分别包含明月和乡的诗歌。



2.扩展后

含两个乡的诗歌排在了同时包含明月、乡的诗歌前面，并且有可能包含近义词的诗歌。


tk
1
2
3
4
5
返回
送乡中故人
方干
少小与君情不疏
听君细话胜家书
如今若到乡中去
道我垂钩不钓鱼
内乡（一作县字） 村路作
白居易
日下风高野路凉
缓驱疲马闇思乡
渭村秋物应如此
枣赤梨红稻穗黄
和微之任校书郎日过三乡
白居易
三乡过日君年几
今日君年五十馀
不独年催身亦变
校书郎变作尚书
送柴侍御
王昌龄
流水通波接武冈
送君不觉有离伤
青山一道同云雨
明月何曾是两乡


tk
1
2
3
4
5
返回
夜期友生（一作贾岛）不至
姚合
忍寒停酒待君来
酒作凌澌火作炭
半夜出门重立望
月明先自下高台
夜过（一作泊）松江渡寄友人
许浑
清露白云明月天
与君齐棹木兰船
南湖风雨一相失
夜泊横塘心渺然
秋月 一作江城秋夜
戎昱
江干入夜杵声秋
百尺疏桐挂斗牛
思苦自看（一作绿）明月苦
人愁不是月华愁
李四仓曹宅夜饮
王昌龄
霜天留后（一作饮）故情欢
银烛金炉夜不寒
欲问吴江别来意（一作处）
青山明月梦中看