

**毕 业 论 文(设 计)**

**论文（设计）题目:**

**中文集句系统设计与实现**

姓 名 **王伟**

学 号 **201300301249**

学 院 **山东大学软件学院**

专 业 **软件工程**

年 级 **2013级**

指导教师 **高晓程**

2017年 5月 10日

**山东大学毕业设计（论文）成绩评定表**

学院：软件学院 专业：软件工程 年级：13级

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学号 | | 201300301249 | | 姓名 | 王伟 | 设计（论文）成绩 |  |
| 设计（论文）题目 | | | 中文集句系统设计与实现 | | | | |
| 指 导 教 师 评 语 |  | | | | | | |
| 评定成绩： 签名： 年 月 日 | | | | | | |
| 评 阅 人 评 语 |  | | | | | | |
| 评定成绩： 签名： 年 月 日 | | | | | | |
| 答 辩 小 组 评 语 |  | | | | | | |
| 答辩成绩： 组长签名： 年 月 日 | | | | | | |

注：设计（论文）成绩=指导教师评定成绩（30%）＋评阅人评定成绩（30%）＋答辩成绩（40%）

**目 录**

[摘 要 1](#_Toc482382469)

[ABSTRACT 2](#_Toc482382470)

[第1章 绪论 3](#_Toc482382471)

[1.1 中文集句系统开发背景 3](#_Toc482382472)

[1.2 国内研究情况 3](#_Toc482382473)

[1.3 遇到的主要问题 4](#_Toc482382474)

[1.4 论文组织结构 4](#_Toc482382475)

[第2章 中文集句系统需求分析 5](#_Toc482382476)

[2.1 中文集句系统综述 5](#_Toc482382477)

[2.1.1 中文集句系统项目背景 5](#_Toc482382478)

[2.1.2 中文集句系统项目介绍 5](#_Toc482382479)

[2.1.3 中文集句系统整体解决方案 5](#_Toc482382480)

[2.2 中文集句系统工作目标以及解决的问题 6](#_Toc482382481)

[2.3 中文集句系统需求描述 6](#_Toc482382482)

[2.3.1 中文集句系统功能性需求 6](#_Toc482382483)

[2.3.2 中文集句系统非功能性需求 7](#_Toc482382484)

[第3章 中文集句系统架构设计 9](#_Toc482382485)

[3.1 中文集句系统设计目标和原则 9](#_Toc482382486)

[3.2 中文集句系统的技术架构设计 9](#_Toc482382487)

[3.2.1 中文集句系统技术架构 9](#_Toc482382488)

[3.3 中文集句系统功能架构 10](#_Toc482382489)

[3.3.1 中文集句系统功能组成 10](#_Toc482382490)

[3.3.2 中文集句系统具体功能流程 11](#_Toc482382491)

[第4章 中文集句系统详细设计 14](#_Toc482382492)

[4.1 中文集句系统静态程序设计 14](#_Toc482382493)

[4.1.1 中文集句系统的数据类 14](#_Toc482382494)

[4.1.2 中文集句系统的网络服务类 17](#_Toc482382495)

[4.1.3 中文集句系统主方法类以及详细功能 19](#_Toc482382496)

[4.2 中文集句系统动态程序设计 22](#_Toc482382497)

[4.3 中文集句系统数据模块部分设计 23](#_Toc482382498)

[4.4 中文集句系统UI设计 26](#_Toc482382499)

[4.4.1 JavaFX架构说明 26](#_Toc482382500)

[4.4.2 中文集句系统UI设计 27](#_Toc482382501)

[第5章 中文集句系统实现与测试 28](#_Toc482382502)

[5.1 中文集句系统实现 28](#_Toc482382503)

[5.1.1 软件界面功能介绍 28](#_Toc482382504)

[5.1.2 中文集句系统的精确&关键字查询界面 28](#_Toc482382505)

[5.1.3 中文集句系统的对偶匹配界面 31](#_Toc482382506)

[5.1.4 中文集句系统的对偶生成界面 32](#_Toc482382507)

[5.2 中文集句系统中文分词模块实现 34](#_Toc482382508)

[5.2.1 标准分词算法 34](#_Toc482382509)

[5.2.2 NLP分词算法[12] 36](#_Toc482382510)

[5.2.3 N最短路径算法[14] 38](#_Toc482382511)

[5.2.4 Dijkstra最短路径算法[15] 39](#_Toc482382512)

[5.3 中文集句系统的测试 40](#_Toc482382513)

[5.3.1 测试的方法论与主要技术 40](#_Toc482382514)

[5.3.2 测试项目与内容 40](#_Toc482382515)

[5.3.3 测试的结论 57](#_Toc482382516)

[第6章 结论 58](#_Toc482382517)

[第7章 致谢 59](#_Toc482382518)

[第8章 参考文献 60](#_Toc482382519)

[附录1 英文原文 61](#_Toc482382520)

[附录2 译文 64](#_Toc482382521)

**中文集句系统设计与实现**

# 摘 要

本文在古诗词的内容上会有简单的描述，然后对于古诗集句软件进行详尽分析，确定功能需求以及非功能需求，对于软件运作流程给以流程图的展示。

在需求分析基础上，进行软件的架构设计。进一步明确使用场景与功能点的关系。在查询模块主要任务是保证查询准确高效，而在集句生成部分，主要利用现有的可行手段，例如开源的中文分词工具对于待对偶诗句进行词性分析，在此基础上，给出相对较优的对偶句。最后，对于使用的开源工具运行机理进行介绍，并对系统，尤其是对偶生成算法的改进方向提出建议。

系统最突出的是将词句匹配以及对偶句诗句生成融合，将客户端以及服务器启动程序结合，增强了程序的易用性以及使用场景的多样性。

**关键字：**中文古诗词；诗词匹配；对偶句生成；中文分词使用

# ABSTRACT

This paper will make a simple description of the ancient poetry. Then, there is an analysis of this ancient poetry software system in detail. After the function requirements and non-function requirements are curtained, there will take the flow chart about software operation process.

The software architecture will based on the demand analysis before. Then, the paper will clarify the relationship between scenes and function points. On the query module, the task is to ensure the query is accurate and efficient. And on the pair generation module, use existing open source project named HanLP, a segmentation tool, to generate sentence pairs with readability according to the word’s property.

At last, the paper will introduce some algorithms used in the project HanLP, and show the direction of system future improvement. System put the client and server into one packet and put the function of pair generating and poetry matching into the other packet which makes the system more reliability and easy-using.

**Keyword:** Chinese poetry; poetry match; Generate pairs; the usage of segmentation

# 绪论

## 中文集句系统开发背景

中文诗句作为传承中国古典文化的钥匙，以其凝练的表述，丰富的含义，优美的意境，在传承数千年中华文明的长河中熠熠生辉。优美的古诗文是全国各地语文教材的重要内容。我们每个人在接受语文教育的同时，都会接触到这精粹的文化。理解诗文内容是锻炼一个人思维的绝佳方式，对诗人的理解，对语境的体会体现了一个人的知识体系的完善程度，根据诗人所处境遇，相似的诗句可能的内涵截然不同。我们只有充分的阅读大量的诗文，才会有一定的能力，模仿创作。

在计算机领域，在弱人工智能领域的机器翻译经过几十年的发展，在中文分词技术上，已经有令人可喜的成果。对于一篇人类的普通文章可以有较高理解程度，并将其转换为其他语言。这一技术基于的是对于语言、语法的理解与建模。

作诗不同于翻译，已经属于强人工智能，模仿人类的作诗行为，是一个崭新的富有挑战的课题。

模仿人的创作是一件很难的事情，尤其是高度凝练的诗句文字，需要使用合适的方式建模模拟这个过程。

国内外的开源工具、开源方法层出不穷，为本软件的技术提供了重要的帮助。自然语言处理方面，HanLP自然语言处理有着良好的解决效率，并且开放了源代码，这让本地实现古诗句的匹配、生成更加具有可读性 (hankcs)。

在数据集方面，网络中已经可以获取大量前人创作的诗文的电子数据，方便进行电子化的处理。

在计算机方面，家用计算机的CPU已经可以胜任一些较为复杂的计算，性能的提高为实现对偶生成创造了可能，并且提高了匹配的速度。

## 国内研究情况

在自然语言的研究上，以百度翻译代表的国内中文处理与Google翻译代表的世界级互联网技术公司的文字处理利用RNN(Recurrent Neural Network)以及LSTM(Long Short-Term Memory)将中文自然语言处理通过大数据量文本推向了一个新的高度。机器人九歌与诗人陈昂联袂出演 《机智过人》，现场作藏头诗引爆全场，其中背后的清华大学的自然语言处理与社会人文计算实验室与搜狗合作提供技术，代表了国内高校研究的最新水平。

## 遇到的主要问题

中文学习中，遇到平仄、对偶模糊查询一般需要使用搜索引擎，并且人工挑选符合需要的结果，然后通过信息进行二次查询，获得诗文、作者。本软件的精确查询与模糊查询可以进行诗文查询与保存，而且数据集成在程序内部，可以离线操作。软件集成服务器模式，根据启动命令不同，可以提供http访问服务，通过设定的request直接获得需要的结果。

需求设计方面，要思考诗句查询可能出现的情况，制定详尽的需求分析表。一个好的需求分析直接决定了一个软件系统的上限，需求设计必须精确而有宏观眼光。

设计方面，考虑到UI部分的简洁实用性，也要对程序的查询架构做好设计，避免占用过多资源。简洁的UI，令人愉快的交互，以及可能轻松愉悦的背景音乐都会让程序的使用体验变得更加富有人文情怀。

在实现方面，首先，定义了较为全面的常用诗句查询规则，对于一般要求的诗句可以较清晰的描述。其次，在词性判断与对偶方面，采用开源的HanLP自然语言处理包中提供的算法，实现更加准确的词性匹配结果。最后，合理运用数据集，通过统计学的知识，让结果符合概率规则，让诗句的可读性更高。

## 论文组织结构

第一章绪论，介绍软件开发的背景，国内外研究现状以及软件解决的问题。第二章需求分析，对软件需求展开分析。第三章架构设计，根据软件需求，设计合理的软件架构，选择合适的软件，设计软件的技术架构与功能架构。第四章详细设计，查询过程建模，查询语句设计，对偶生成方法设计，网络通信设计，并简单介绍使用的中文分词方法。第五章实现与测试，使用单元测试检测各方法的执行效果，使用白盒测试整个软件查询流程。第六章结语，对软件进行总结，并提出改进方法以及措施。

# 中文集句系统需求分析

## 2.1 中文集句系统综述

### 中文集句系统项目背景

查询古诗词不便，模糊查询与精确查询需要连接网络并且反复查询，需要一款可以按照一定模式进行精确查找的软件，对于古诗的学习，查询有较好的效果。中文分词软件的准确性不断增加，开源项目增多而且成果显著。利用现有的手段以及个人计算机，可以尝试解决古诗文查询以及匹配生成的问题。

### 中文集句系统项目介绍

中文集句系统是为了更方便快捷的查询特定类型古诗词而开发的工具，通过查询语句对于古诗约束而针对性的查询出相关结果，并导出结果。对于古诗词的初学者，尤其是平仄韵律学习者，以及古诗词爱好者和有诗词查询需求的人，有较大的帮助作用。并且通过适当的启动命令可以担当小型查询服务器的任务，便于以后的功能拓展与业务拓展。在使用系统的时候，需要仔细阅读相关的程序说明，合理使用界面提供的功能。该项目已经在github开源，欢迎直接查看源码，批评与指正。

### 中文集句系统整体解决方案

系统采用模块化系统组织，由用户交互界面模块、逻辑处理模块以及数据加载模块组成。其中，逻辑处理模块增加了处理网络请求的分模块。整个软件免安装，可以直接使用jar包启动，数据集内嵌程序中，不必考虑网络情况对于软件系统的影响。网络部分采用HTTP的Post与Get获取数据与返回结果，客户端demo代码也集成在jar包中。这种解决方案，虽然在计算能力与查询速度上带来的架构提升不大，但是十分适合在轻量级的电脑上运行。程序的内从占用与CPU使用不高，可以保证软件对于硬件的要求，同时可以为移植到移动端打下基础。

下面是程序的结构图：



图 2‑1程序结构图

## 中文集句系统工作目标以及解决的问题

本系统目标：可以完整执行用户关于诗句的完整查询、结果预览、结果保存、对偶诗句生成、对偶诗句匹配等功能，形成可安装使用的完整程序。

本系统解决的问题：

1. 诗句的具体约束查询，准确而快速
2. 对偶句生成有意义的诗句

其他问题，JavaFX的UI界面使用，多线程操作，XML数据内存化技术，匹配优化等。

## 中文集句系统需求描述

### 中文集句系统功能性需求

1. 系统涉及的诗句查询匹配

本系统匹配的诗句有几大约束部分。其中，主要使用的约束为：诗人约束、通配符、叠字约束、平仄约束、任选字符约束、声调约束（平仄约束）、拼音约束、或约束等。对于以上的约束，使用系统查询需要自定义一种表达方式来区分不同约束，各种约束见图2-2：

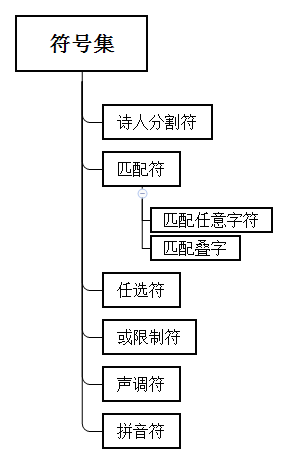


图 2‑2 各字符约束

根据查询的要求不同，将查询分为了两个大类。

第一类，精确到逐字要求且字数有限制。

查询样例（1），用户希望查询李白或者杜甫五言诗句中提到“紫”字，且该字出现在诗句的第1个字。对应的精确查询语句为：

李白|杜甫&紫\*\*\*\*

第二类，查询的诗句中的字数不确定。

查询样例（2），查询王维诗中提到“鸡黍”的诗句，查询关键字诗句为：

王维&鸡黍

通过规则对于诗句的描述，可以快速定位到相关诗句，而充分使用上述字符可以满足使用者对于诗句查询的要求。

### 中文集句系统非功能性需求

1. 程序可以在Windows与Linux系统通用，该程序需要面对的使用者并不限制其操作系统的多样性。
2. 便于部署，提供网络接口方便拓展其他界面的功能，可以提供多样化的服务。
3. 查询的速度不可以超过90s，对于封闭数据集合，典型的7言诗句，所有的诗句全部吻合的时间为最高时间。
4. 在对偶生成期间，速度根据字数差异，典型为生成7言小于100s。
5. 对于异常数据有容错性，对于无属性的生僻字不能导致查询的中断。
6. 用户交互，确保全功能可以正常使用。
7. 生成的文件有一定可读性，需加入一定排版。

# 中文集句系统架构设计

## 中文集句系统设计目标和原则

中文集句系统的设计目标是满足需求分析的需求，并设计出可拓展的系统。

中文集句系统的设计原则主要体现在模块化思想、多服务模式和系统可拓展三个方面。

模块化思想，每个模块之间分清功能以及确定接口设计，模块内部实现全部逻辑，不能相互干涉使用模块内部方法。

多服务模式，所有的业务逻辑封装在jar包中，根据启动命令不同，启动不同的服务模式。

系统拓展方面，开启网络服务模式，只需要进行HTTP通信，就可以使用Client模式的绝大部分功能。所以，可以拓展到WEB页面的服务以及手机端。

## 中文集句系统的技术架构设计

### 中文集句系统技术架构

采用JavaFX[4]进行界面UI设计交互，Java JDK 1.8.0进行主程序编写，XML进行数据的组织以及标准化。使用jetty作为Web服务器的依赖包，提供Server服务。使用log4j作为系统日志生成组件。

JavaFX，将界面展示与逻辑处理分开，摆脱了界面代码与逻辑代码混杂的Java界面设计模式，而且支持CSS等，让界面设计更加简洁而专一。

XML文档有着强大的表述能力，规范的读写控制，作为轻量级的数据存储单位有着比数据库更加快速的优点。

Jetty框架，提供轻量级的JavaWeb框架，比TomCat更加轻量级，且支持的jar包少，通过Maven引用简单快捷。

Log4j是Apache开发的日志输出组件，对于监控运行的服务器端Java程序的运行状况有着十分明显的作用。

具体详细结构见图3-1：

图 3‑1 详细技术架构图

## 中文集句系统功能架构

### 中文集句系统功能组成



图 3‑2 中文集句系统功能组成图

主要功能分为三个：

1. 诗句匹配

对与诗句是运用规则进行精确匹配；使用关键字进行模糊匹配，匹配中运用部分精确匹配规则。精确匹配，是已知诗句个数，进行匹配时，首先排除数据集中不满足诗句长度的诗句，从而达到提高效率的目的。模糊匹配，是对应一首诗中不知道诗句长度，而需要查询所有诗句才能判断是否匹配成功，但是结果丰富，对于特定场景有很强的效果。

1. 对偶匹配

对于输入的一句诗句（包括自己创作的），对于数据库中的诗句进行对偶匹配，返回符合对偶条件的诗句。对偶匹配的规则有很多，根据对于诗词格律概要[2]的研读，对偶对于平仄、词性有一定要求，在众多的对偶规则中，本文选取基本对偶规则进行对偶判断。

1. 对偶生成

对于输入的诗句，通过一定算法返回合适的对偶诗句。对偶输入不仅要输入待匹配诗句，还需要输入生成个数，便于程序有限词的生成结果，程序本身不能评价生成诗句的好坏，所以需要人工对于生成结构进行整理分析。

此外，服务器模式，实现HTTP通信使用上述三个功能并返回结果。对于网络客户端的代码，可以参考程序内的DEMO程序，在网页或者手机端可是同时调用接口，实现功能拓展。

### 中文集句系统具体功能流程

功能流程图见3-3，3-4：



图 3‑3 匹配诗句功能流程图

3-2匹配流程适用于精确匹配与关键字匹配，以及对偶匹配。匹配设计与第四章详细介绍。



图 3‑4 对偶句生成流程图

对偶句生成的分词，以及获得对偶词将由第四章详细展开。对偶句的关键逻辑在于对偶词的生成，以及中文分词的准确程度。

# 中文集句系统详细设计

经过需求分析和架构设计，逐步分析了中文集句系统涉及的需求和架构流程。本章在第三章的基础上，分析系统的模型结构和使用的算法以及诗句匹配的详细过程。详细设计分为静态设计、动态设计以及数据组织管理。

## 中文集句系统静态程序设计

### 中文集句系统的数据类

基础文字类wordClass.java，负责存储诗的最基本元素单词。



图 4‑1 wordClass.java类图

一个单词，属性有word字形、pinYin读音、pingZe平仄、yunbu在平水韵中所在对应韵部、kind平仄以及在全诗词中出现的次数。因为有三种主要功能所需要的wordClass需要承担不同的责任，所以我适应多态，不同的构造函数生成的word可以满足不同需要同时减少所有参数输入的问题，这部分将在具体情况下展示。

与wordClass相似，诗句使用tangClass进行存储。



图 4‑2 tangCass.java类图

tangClass中，author为作者，dynasty为朝代，context为诗文的String对象，包括换行与标点符号，title为题目，pairs为对偶生成结果存放位置，pairsWegiht为生成结果权重值（第一代对偶生成算法使用），关于对偶生成策略，在之后详细介绍。

枚举类operation.java作用于服务器模式，http请求中的请求转化为业务请求。



图 4‑3 operation.java 类图

数据加载部分，由六种数据源组成，分别是诗句集合、平水韵表、作者集合、字典集、字分类集合以及诗句集字统计后形成的表。

1. 诗句集合，诗词查询的核心，诗句全部集中于该集合。
2. 平水韵表，根据诗词格律概要[2]中的描述，现存的最早诗韵是《广韵》，其前身为《唐韵》，而《唐韵》的前身为《切韵》，《广韵》共206韵，唐初许敬宗等奏议合并中临近韵。宋淳祐年间，江北平水人刘渊著《壬子新刊礼部韵略》，合并206韵至107韵。清代改称“平水韵”为“佩文诗韵”，合并为106韵，唐诗所用韵律实际使用平水韵。本程序主要解决唐诗及以后的研究。所以平水韵十分适合作为程序使用的韵律表。
3. 作者集合，是诗词集合的子集，主要作用是在查询过程中加快对于有诗人限制的匹配工作
4. 字典集，是拼音、字形的基本依据。收录了常用的汉字，并不含生僻字。
5. 字分类集合，这个集合感谢我的导师高晓程老师，他在繁忙的教学间隔中，根据自己的查询、以及自己的判断，整理了一个简易的字分类集合。本集合也是第一对偶生成算法的基础。基本思想是，用一个32位的数字表示汉字的属性，每一位表示一种属性，例如名词、动词、介词等。基本上分了26类，使用的时候，用两个汉字的对应数字相与如果相同属性位置均为1，则认为两个字的词性是匹配的。从数字角度上考虑，若两数字完全不匹配，则结果为0，数字越大，两汉字越匹配。这种生成对偶字的方法并不考虑词对于对偶的影响。同时，在对偶匹配中，也是用到了这个数据集。
6. 诗句集字统计表，是根据诗句集，将所有出现的汉字分类，并统计其出现次数的表格，因为第二种匹配算法使用到汉字出现概率，所以需要这项数据，这项数据对于固定集合也是固定的，所以固定成表加快运行速度。

由于这六个数据集有较大的相似性，结构很相似，所以我以及字典集为例：wordSource.java



图 4‑4 wordSource.java 类图

Logger对象为log4j包需要打印日志定位Class。ArrayList<wordClass>为数据集转换的java内存对象，也是程序搜索时取得数据对象。构造方法wordSourc()，将XML文件转化为ArrayList<wordClass>对象，getWordClassArrayList()返回生成的结果集。

其他的数据集合类如：tangSource.java，kindSource.java均是如此构造可以将该类看为接口类来理解。

### 中文集句系统的网络服务类

网络部分，主要使用的类netServer.java



图 4‑5 netServer.java 类图

netServer负责启动服务器线程，hnadler处理每一个http请求，对于合法请求，经过调用主方法类，获得结果后返回。

httpRequest结构

|  |  |
| --- | --- |
| Header | 可输入项 |
| OPERATION | isSearch，possibleSearch，pairMatch中任意一项 |
| INPUT | 实际查询语句，例如“李白|杜甫&桃花潭\*\*\*” |
| URL | 启动Server服务器的主机ip：port |

而客户端可以通过构造event对象实现对于客户端的response值的读取，代码如下

CloseableHttpResponse response = httpclient.execute(request);

//解析response

InputStream bis = response.getEntity().getContent();

ObjectInputStream ois = new ObjectInputStream(bis);

event e = (event) ois.readObject();

logger.info(e.toString());

int status = response.getStatusLine().getStatusCode();

logger.info("返回 "+status);

System.out.println(status);

httpclient.close();

event是一个构造的网络传输对象，并且继承了Serializable接口。

public class event implements Serializable{

private static final long serialVersionUID = 3584042093326873203L;

public ArrayList<tangClass> getResult() {

return result;

}

public void setResult(ArrayList<tangClass> result) {

this.result = result;

}

public String getInputString() {

return inputString;

}

public void setInputString(String inputString) {

this.inputString = inputString;

}

public operation getOp() {

return op;

}

public void setOp(operation op) {

this.op = op;

}

private ArrayList<tangClass> result;

private String inputString = "";

public operation op;

public event(operation op,String input){

this.setOp(op);

this.setInputString(inputString);

}

public String toString(){

String out = "";

for (int i = 0; i < result.size(); i++) {

out = out + "\n" + result.get(i).toString();

}

return out;

}

}

### 中文集句系统主方法类以及详细功能

主方法类analyze.java



图 4‑6 analyze.java 详细类图

该工具类实现了查询的全部功能，主要入口函数为

Match(String),PMatch(String),pairMatch(String),getPair(String),对应精确匹配、关键字匹配、对偶匹配以及对偶生成。

1. Match(String)

首先判断输入合法，使用getAuthor()对有诗人限制的提取，并查询诗人集合，无结果直接返回，有结果进入下一步。

调用isMatch(String,String)，左侧为待匹配字符，右侧为诗句的Context属性。isMatch中，对于诗句Context中的每一句中的每一个汉字，都使用judge(String,String,String)判断是否与输入的匹配字符串吻合。judge中，第一个参数为匹配规则字符，第二个为来源于诗句Context中的待匹配汉字，第三个参数为上一个匹配的汉字。例如，juage的输入为”\*”，”白”，“”。那么根据匹配字符规则，\*对应任意汉字，那么返回true。输入为“a”，“黑”，“白”。那么根据规则，希望查询叠字，但是本字为黑，上一个汉字为白，所以返回false。

judge为最基本的规则判断方法，处理了基本所有的规则符号。具体代码参考附录中的analyze代码描述。

1. PMatch(String)

关键字查找的思路与精确查找一直一致，关键是两个单字之间空有复数个汉字时，使用第一种按序列匹配不能解决问题。

首先也是对输入检查，调用possibleMatch(String,String)，这个方法与isMatch()对应，在这个方法中，首先对于输入的查询规则进行建模，找到能影响规则的字符位置，这些字符有：汉字，[]，调用getKeyIntArray(String),获得关键字数组。对于每一次的诗句循环，建立一个初始化全0的二维int数组isFound[][]，存放key值与对应的其他字符的匹配情况。当key值字符在其他位置找到对应字时，在isFound的key值所在数组位置a，匹配到汉字的位置b标记为isFound[a][b] = 1。最后，遍历一句话的isFound数组，如果有一处为1，那么就可以判断这句话匹配成功，返回这首诗。这就完成了关键字搜索部分的关键代码段。

1. pairMatch(String)

这个对偶匹配方法使用的就是字分类集合，对输入的每一个字与诗句集中的相同大小的诗句进行一次对偶计算。由于分类集中汉字个数2177，远小于诗词集的8116。所以，对偶的匹配效果并不理想。这也是制约对偶诗句生成的主要问题。

1. getPair(String)

主要思路是首先获取输入字符串的中文分词，然后对于每一个输入的中文分词，生成对应的对偶词。对于诗词集合中的文字统计表，我们已经在上文中介绍并且实例化到了程序中。生成随机数R，对于诗词集的每一个字出现的频度A，总字数为M。我们在wordClass有记录count属性。所以，我们可以得到依据诗词汉字出现频率来随机生成的一个较为可信的随机汉字而不汉字，当

且，

认为该序列的汉字为对应生成的可信汉字。

我们得到单个汉字之后，按照相同方法获得下一个汉字，直到达到分词结果的字数。然后对生成的汉字使用分词工具，判断词性，若与输入分词的对应词组词性相同，就直接填入结果中。

例如：输入语句为“月涌大荒流”

首先中文分词，分为[月/q，涌/v,大/a,荒/ng,流/v]



然后对应生成“天如清寝尽”。

## 中文集句系统动态程序设计

系统动态设计是通过流程图、时序图展示的。中文集句软件主要是对于输入的查询、匹配。下面是对于系统的部分功能流程描述。



图 4‑7 UML活动图

在结果界面，可以选择保存文件，也可以点击单个诗句来进行保存操作，程序开始界面还有部分非必要功能，在这里不再赘述。

## 中文集句系统数据模块部分设计

数据模块的思想是将待查询数据直接载入内存，加快程序判断速度。在程序中，使用ArrayList作为基本类型wordClass与tangClass的组织对象，虽然六种数据源的数据不尽相同，但是最终都可以归纳到上述两个基本类的对象的属性之中。区别仅在于，数据加载程序需要根据不同的XML文件进行调整。

以下是不同文件的实际内容截取：

1. fullshici.xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<唐诗三百首 xmlns="http://tempuri.org/tangshi.xsd">

<唐诗>

<题目>少年子</题目>

<作者>李白</作者>

<正文>青云年少子，挟弹章台左。鞍马四边开，突如流星过。金丸落飞鸟，夜入琼楼卧。夷齐是何人，独守西山饿。</正文>

<注解 />

</唐诗>

</唐诗三百首>

1. pingzezidian.xml

<?xml version="1.0" encoding="UTF-8"?>

<dataroot xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" generated="2017-03-10T10:08:22">

<pingzezidian>

<字符>一</字符>

<读音>yi2 yi4 yi1</读音>

<平仄>2</平仄>

</pingzezidian>

</dataroot>

1. authors.xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<NewDataSet>

<authors>

<name>李清照</name>

</authors>

<authors>

<name>李世民</name>

</authors>

</NewDataSet>

1. zifenlei.xml

<?xml version="1.0" encoding="UTF-8"?>

<dataroot xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="zifenlei.xsd" generated="2017-03-10T10:05:19">

<zifenlei>

<zi>打</zi>

<fenlei>2</fenlei>

</zifenlei>

</dataroot>

1. pingze.xml

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<root>

<字符>东<上平去入>0</上平去入><韵部>1</韵部></字符>

</root>

1. wordCountList.xml

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<root>

<字符>川<出现次数>2332</出现次数></字符>

</root>

在主分析类中，构造函数中自动加载上述六种数据源。下边是wordClass与tangClass的详细属性。

wordClass

|  |  |  |
| --- | --- | --- |
|  | 类型 | 说明 |
| word | String | 汉字字符 |
| pinYin | String[] | 拼音最后一位是读音，支持多音字 |
| pingZe | int | 有0，1，2，3三种对应平，上去入（仄） |
| yunbu | int | 平水韵的韵部[[1]](#footnote-1) |
| kind | int | 对应32位的词性分类，每一位代表一个属性 |
| count | int | 出现在诗句集中的次数 |

tangClass

|  |  |  |
| --- | --- | --- |
|  | 类型 | 说明 |
| author | String | 作者 |
| dynasty | String | 朝代 |
| context | String | 诗文，包括标点、换行符等 |
| title | String | 题目 |
| pairs | String | 对偶匹配成功的诗句 |
| pairsWeight | int | 匹配度，成功配对的诗句所有汉字权重累加 |

## 中文集句系统UI设计

由于使用Java作为开发语言，界面的开发是一个相对困难的问题。Java Swing是Java自带的界面开发框架，但是版本老旧，代码逻辑与界面代码混杂，不便于开发、维护工作。JavaFX是Oracle内置于Java7版本以及Java7的图形开发框架。经过简单的学习，就可以设计出外观简洁、代码简洁的UI部分，并且逻辑块与界面代码不会相互干扰，增加可读性。JavaFX可以通过网上的中文JavaFX网站[5]学习，上手方便快捷。

### JavaFX架构说明

JavaFX是一个Orcale公司为了改进Java的图形界面而开发设计内置于Java8中的图形和多媒体处理工具包的集合，JavaFx可以让开发人员来设计、创建客户端的界面程序，并且和Java一样跨平台[6]。

一般的JavaFX分为三个部分，fxml描述UI文件、JavaFX Application、JavaFX Controller。

1. Fxml，描述界面设计的文件，作为附加资源被Application启动时载入。

下面是一个Fxml文件的部分内容。

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.scene.text.Font?>

<BorderPane fx:id="bigBorderPane" xmlns="http://javafx.com/javafx/8.0.101" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.Davi.poems.gui.Controller">

<top></top>

</BorderPane>

1. JavaFX Application

Application是一个抽象类，规定了start()方法，用于创建最初窗口，一个实际使用的界面继承Application，创建Controller对象并且加载fxml文件。

自定义方法welcome,加载Controller以及fxml文件。

Public void welcome(){

try {

Controller controller = replaceSceneContent("/welcome.fxml");

controller.setApp(this);

} catch (Exception e) {

e.printStackTrace();

}

}");

1. JavaFX Controller

Controller是真正的逻辑处理部分。之中，fxml文件描述的组件的fx：id在Controller中使用关键字 @FXML 引用。每一个方法必须在fxml中填写才可以生效。

Controller中的一个方法。

@FXML public void playOrStopBGM(){…}

Fxml文件中的内容对应。

<MenuItem mnemonicParsing="false" onAction="#playOrStopBGM" text="关闭&amp;开启音乐" />

### 中文集句系统UI设计

根据需求分析部分，主要的查询界面有3个，分别负责精确&关键字查询、对偶生成、对偶匹配。

所以采取的对应设计为一个主类Main.java继承Application。在之中设计连同欢迎界面在内的4个fxml文件。每一次切换Controller的场景上下文来实现切换页面的操作。这样可以将处理逻辑集中在一个Controller中，便于实现以及维护。

# 中文集句系统实现与测试

## 中文集句系统实现

### 软件界面功能介绍

中文集句系统欢迎界面实现如下图5-1所示：



图 5‑1欢迎界面

点击开始集句之旅进入第一个查询界面，也可以点击文件，可选择三种不同的功能。附带诗句为本人拙作。

### 中文集句系统的精确&关键字查询界面

精确&关键字查询界面图5-2：

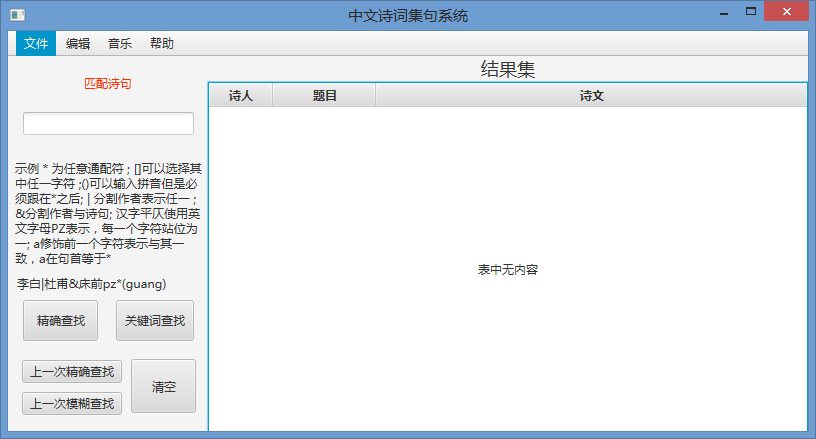


图 5‑2 精确查询&关键字查询界面

输入框下方简单介绍了查询的匹配规则，并给出简单的示例。

将文本输入文本输入框，点击精确查找，在右侧结果集返回结果图5-3：

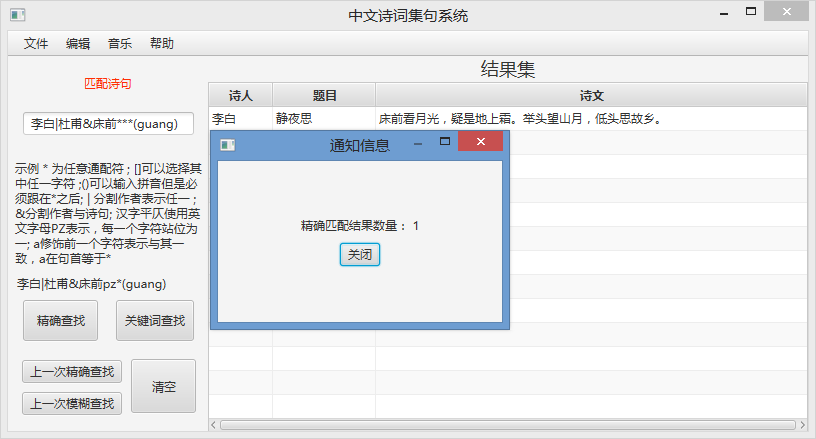


图 5‑3 精确查询使用图

此时，点击右侧结果集图5-4：

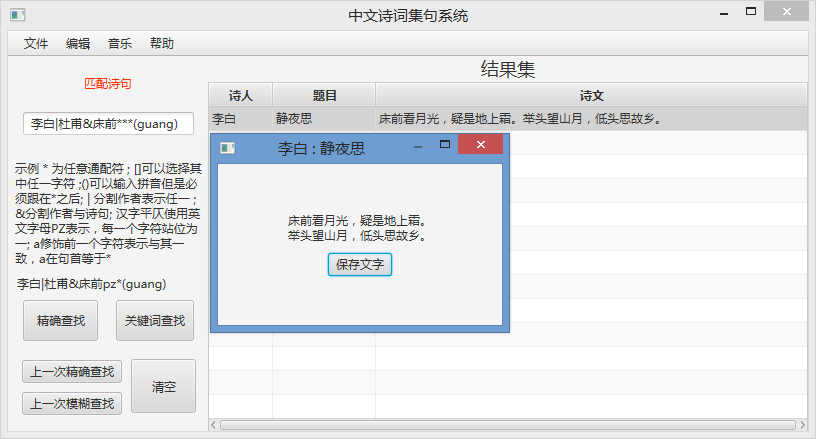


图 5‑4 结果查看

这是诗句的详细内容，并且附加了保存文本选项，可以选择保存目录并保存图5-5：

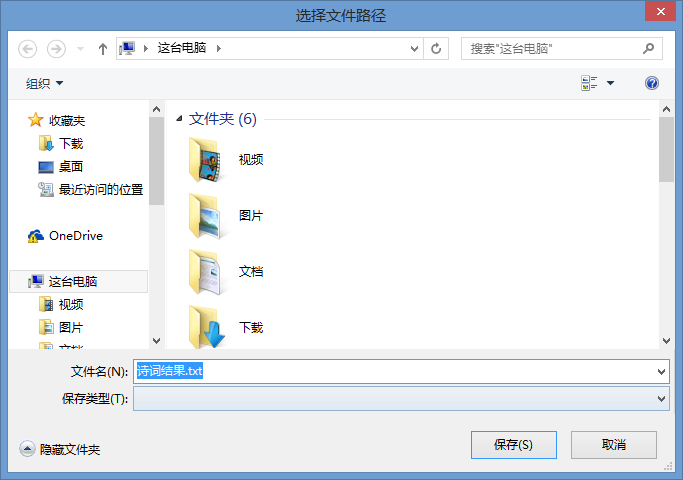


图 5‑5 保存查询结果

还可以点击菜单栏编辑选项，保存结果，这是用于批量保存文件的方法。保存当前的结果集内容。

### 中文集句系统的对偶匹配界面

对偶匹配，点击文件-对偶匹配图5-6：

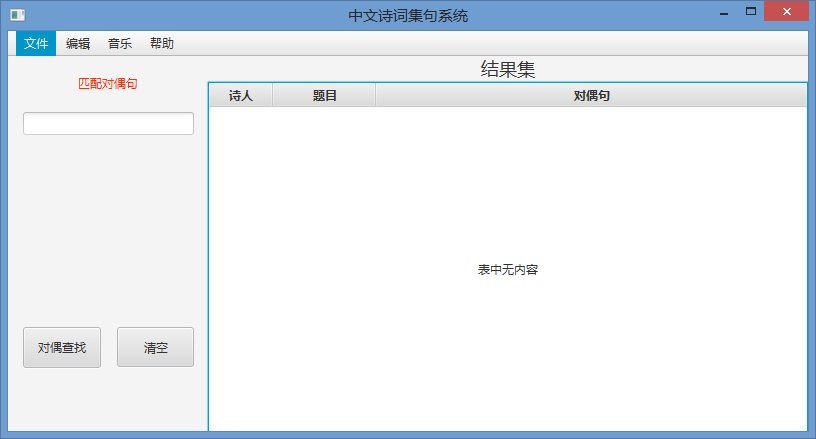


图 5‑6 对偶匹配界面

对偶匹配的方法与精确匹配类似，可以对诗人进行限制，但是一般为了找到合适的对偶句，不做限制，进行查询5-7：

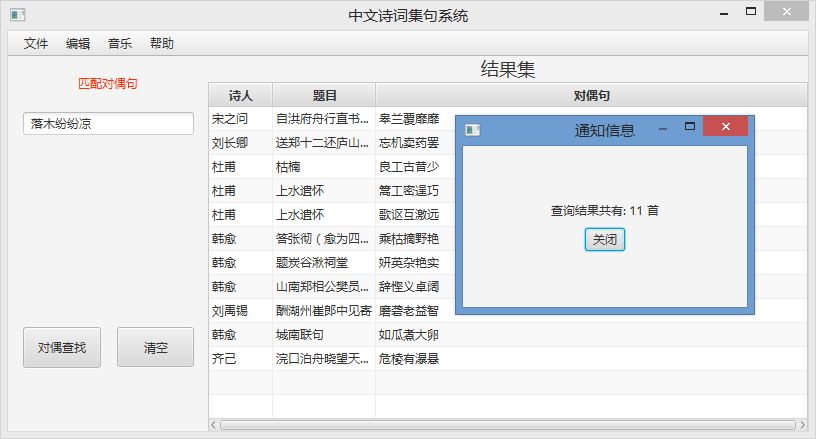


图 5‑7 对偶查询结果

查询速度较慢，因为使用的词性字典并不权威，所以结果不做置信的保证。

### 中文集句系统的对偶生成界面

对偶生成，点击文件-对偶生成，图5-8：

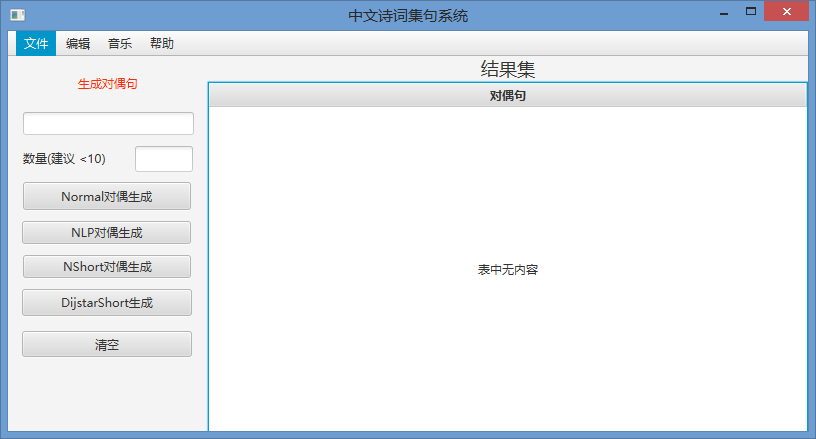


图 5‑8 对偶生成界面

软件集成4种中文分词算法，所以对偶生成的结果有不同，算法的介绍将会在5.2说明。

输入待生成的对偶句上句，图5-9：



图 5‑9 normal分词的对偶生成结果

HanLP的NormalSegment是作为默认调用的分析方法，结果如上图。

其他方法直接截图，仅有使用分词算法不同。



图 5‑10 NLP分词的对偶生成结果



图 5‑11 N-最短路分词的对偶生成结果



图 5‑12 Dijkstra最短路径分词的对偶生成结果

## 中文集句系统中文分词模块实现

本软件使用的标准分词算法、NLP（自然语言处理）分词算法、N-最短路径分词算法以及Dijkstra最短路径算法分词将在本部分做介绍并分析算法异同。有关内容可以参考[1]，网站对于HanLP的使用方法以及设计。

### 标准分词算法

生成词图，分词系统需要字典数据。词图，指句子中所有可能组成的词语的图。若有两个汉字A，B。其中，A是B的前驱，B是A的后继。那么A，B两个汉字间有一条词通路径（A，B）。一个词可以有N前驱或者后继。由A，B这样的字以及他们的前驱与后继构成的图，称为词图[7]。

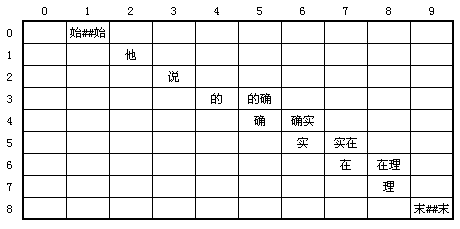


图 5‑13 稀疏二维矩阵

其中，建立点A，B的联系，找到后继前驱的算法有两种。

1. 使用 DynamicArray

直观上，建立一个二维的表，从一个词最左侧汉字序列作为行，最右侧汉字序列作为列，就可以表达n个汉字之间的关系。但是会出现很多无效格，占用内存。 DynamicArray的每个节点都包含词语的行编号(row)与行编号(col)也就是词的起止位置。

上图中，列为n的词可以与行为n的词进行组合。例如，“实在”来源于（5，7）也就是从第五字符后到第七字符，而“实在”的col为7，那么与它一起计算平滑值的词就在row为7的“理”。连接词形成边使用上面的规则。在查询和便利的时候，需要逐个比对col与row，复杂度为O(n)。

1. 使用quick offset

使用offset将相同的词写到同一行，建立一个一维数组，而数组内的每一个元素是一个单链表。

1.始##始

2.他

3.说

4.的/的确

5.确/确实

6.实/实在

7.在/在理

8.理

末##末

当查找下一个词的时候，以“实在”为例，实在位于6号位置，而实在的词语长度是2，所以下一个词的位置是2 + 6 = 8，8号位置。

所以当查询和插入的时间均为O(1)。

在生成词图后，就要进行词性标注工作。词性标注工作涉及面广，技术难度复杂，所以下文将引用成型文献进行解释。

词性标注（Part-of-Speech tagging 或POS tagging)，又称词类标注或者简称标注，是指为分词结果中的每个单词标注一个正确的词性的程序，也即确定每个词是名词、动词、形容词或其他词性的过程。在汉语中，词性标注比较简单，因为汉语词汇词性多变的情况比较少见，大多词语只有一个词性，或者出现频次最高的词性远远高于第二位的词性。据说，只需选取最高频词性，即可实现80%准确率的中文词性标注程序[8]。

HanLP中使用了一阶隐马模型[9]，在这个隐马尔可夫模型中，隐状态是词性，显状态是单词。

HanLP训练语料采用了2014人民日报切分语料。

汉语词的词性频数词典

分别统计所有单词的不同词性的出现频数，记录在词典的词语属性中，得到汉语词的词性频数词典（局部）如下：

爱 v 3622 vn 598

爱因斯坦 nrf 20

爱国 a 178

～

统计每个标签的转移频率制作转移矩阵。

利用求出的转移矩阵和核心词典词频数据可以计算出HMM（隐马尔可夫模型）中的初始概率、转移概率、发射概率，进而完成求解，关于维特比算法和实现请参考通用维特比算法的Java实现[11]。

关于词性标注部分，请参考2014年人民日报切分语料。

通过使用HMM模型，以及训练集的初步训练，得到了HanLP的标准分词算法，也是中文集句系统使用的基本算法之一。

### NLP分词算法[12]

NLP（自然语言处理）技术，目前取得了丰硕的成果，并且广泛应用于基于文档的信息检索、信息抽取与机器翻译等领域。目前的NLP在于数据检索上的应用可以分为两个层次，即词语层和上词语层。在词语层这个方面，NLP应用于自动分词识别内容词与功能词、识别复合短语与专有名词以及未收录名词。具体的应用为原形化或词干提取算法、机器可读字典开发、词语索引以及消除词语歧义和异意。而在上词语层，NLP应用到了概念表示、句法分析以及语义分析。

NLP分词算法主要用到的是句法分析[13]。在自然语言中，语法规则是必需的存在，用于规范语言让语言有规律可循，一个句子中可能会出现主语、谓语、宾语。自然语言处理中的句法分析就显得比较重要。句法分析主要解决两个问题，第一是句法在计算机的存储和描述，第二是句法分析算法。所以，解决第一个问题，使用句法树来进行存储。

假设，使用S作为句子，N，V，P代表名词、动词与介词，p代表短语（Np，Vp，Pp分别代表名词短语、动词短语、介词短语）。那么一句话就可以使用上述的描述作为计算机内的句法存储结构。

例句：Jinan is located in Shandong Province

S：Jinan is located in Shandong Province

N：Jinan Shandong Province

V：is located

P：in

(S (Np (N Jinan)) (Vp (V is) (Vp (V located) (Pp (P in) (Np (N Shandong)(N Province))))))

在解决存储问题后，解决第二个问题是句法分析算法。

采用CFG(Context-Free Grammar)，上下文无关语法算法。为了简单的得到一颗语法树，定义语法如下：

1）N表示一组非叶子节点的标注，例如{S、Np、Vp、N...}

2）Σ表示一组叶子结点的标注，例如{Jinan、located...}

3）R表示一组规则，每条规则可以表示为X->Y1Y2...Yn，X∈N，Yi∈(N∪Σ)

4）S表示语法树开始的标注

N = {S,Np,Vp,Pp,V,N,P,Vi,Vt,NN,IN}

S = S

Σ= {Jinan,is,located,in,Shandong,province }

R={s->NP,VP;VP->Vi,Vt,Vp,Np,Pp;Np->Np,NN,Pp;Pp->IN,NP}

由于句法分析的多重意义的性性质，所以在一个句子中可能会获得多个语法树，但是正确的语法树往往仅是一个，所以采用的一种常见的解决方法就是PCFG(Probabilistic Context-Free Grammar)，概率分布的上下文无关语法。

为了让语法树更加清晰，需要统计语法规则，对句子的语法规则增加置信程度，倾向于更有可信度的语法树，R规则内部，每一个句法规则都增加置信程度，例如：

S->Np,Vp 1.0

Vp->Vt 0.4

Vp->Vi 0.4

Vp->Vp 0.2

Np->NN 0.3

Np->Np,Pp 0.7

Pp->P,Np 1.0

而对于生成的语法树，对于每一个组成的规则的置信概率的乘积作为语法树置信程度。而同时出现多颗语法树时，有理由相信出现的可信度最大的语法树就是最有可能的的语法树结果。

随后，利用足够大训练样本对于模型进行训练，PCFG中的参数经过训练调整到较符合训练集的程度。

1）统计出语料库中所有的N与Σ；

2）利用语料库中的所有规则作为集合R；

3）针对每个规则A -> B，从语料库中估算置信程度p(x) = p(A -> B) / p(A)；

因为多语法树在解析上存在问题，所以在CFG的定义上，定义二叉语法的语法格式。要求每条规则只能是X -> Y1 Y2或者X -> Y的格式。实际上二叉语法格式保证产生的语法树总是二叉树语法树的格式，同时普通语法树可以转化成二叉语法格式。

当训练完毕，拥有PCFG的模型完整参数，且语法转换为二叉树语法结构时，输入S=X1，X2，…，Xn计算句子对应语法树。

由此基本的NLP的语句分析算法设计完毕，但是算法存在一些缺点：

1. 词法信息依赖外部的信息提供，训练集的丰富程度决定最终结果
2. 连续介词的处理在训练集中难以训练
3. 训练集对于使用环境的限制与要求

### N最短路径算法[14]

N最短路需要记录从哪一个点到哪一个点，花费是多少，采用的数据结构是PreNode节点。

一个PreNode节点可能出现的内容有

1. 上一个N最短节点的节点名
2. N最短路径的权重
3. 记录第N最短的index值

N最短路径的基本思想是使用Dijkstra算法，以1-最短路径为例，求到下一个节点的最短路，沿最短路径前进至某个节点时，检查到下一个节点是否有其他路可以警告过，如果有的话，从发现的新的路径中选择一条路径，同时将所有最短路径加入PreNode而不是仅仅加入一条路径。而N最短就是第N短路径，维护N种PreNode，内容是第N短路径的上一个点的信息。

所以，在拥有词语训练集的情况下，我们会拥有一个某词语到某词语的权重（概率分布），当输入一个字符串的时候，我们首先生成词图，并且去掉不存在的词语，然后将剩余词放入N最短路径算法模型中。

例如：

S = “商品和服务”

可以分解为：

“#开始”“商”，“商品”，“品”，“和”，“和服”，“服”，“服务”，“#结束”几个单词（“品和”未查询成功）,且从0开始依次编号到8

可能出现的情况

0-1 “#开始”to“商”

0-2 “#开始”to“商品”

1-3 “商”to“品”

2-4 “商品”to“和”

3-4（不通）“品”to“和”

2-5 “商品”to“和服”

3-5 “品”to“和服”

4-6 “和”to“服务”

根据词语的训练集，我们可以获得每个字到下一字的出现次数（或者概率），所以根据N最短路径的算法，只需要在每一个边上加上权值（概率）就可以求出对应的分词情况。

### Dijkstra最短路径算法[15]

Dijkstra是一个十分经典的单源最短路径求解算法，在《算法导论》中作为经典算法进行讲解，本文作为回顾，简单介绍一下Dijkstra的核心思想。

Dijkstra解决的是一个有向的带权图（非负）的最短路径求解的问题。对于分词的问题来说，只要通过训练集获得词语相关权重，查询输入字句的最短路径和5.2.3 N最短路径算法中N = 1的情况一致。

Dijkstra算法涉及到松弛技术。对于每一个到原点S的顶点V，都维护一个d[v]，含义是到原点S的估计最短距离，对于一个权值w(u,v)的边来讲，d[u]与d[v]的关系为

1. d[v] >= d[u] + w(u,v)
2. d[v] < d[u] + w(u,v)

对于第一种情况，说明v点的最短路径不通过(u,v)；而第二种情况，需要松弛边w(u,v),给d[v]赋予新的权重。

Dijkstra算法反复选取从顶点开始的所有与当前最短路径节点最近的节点（通过松弛当前节点的邻接节点选出最小的节点之一），最后得到一条最优解。

在算法进行过过程中，只能得到一个关于当前输入语句的解，这是与1-最短路径的主要区别之一，但Dijkstra算法因为随机选择而使得计算量变小，计算速度加快，这是算法优点之一。

## 中文集句系统的测试

### 测试的方法论与主要技术

软件测试，作为保证软件工程质量的关键性步骤，对于软件的水平起到了举足轻重的作用。测试的作用有两个，其一，是对质量做出判断；其二是发现存在的问题。根据软件工程的方法论，本系统的开发已经经历了需求分析、概要设计、详细设计、代码实现。接下来，本部分对系统采用3种不同的测试技术，用来测试软件的质量水平。

测试主要使用方法有，功能测试、结构测试。

功能测试又称黑盒测试，核心内容是给定输入获得正确输出，即程序正确。

结构测试又称白盒测试，核心是已知程序结构，在输入数据的基础上，可以准确判断程序的运行情况以及定位出现的问题。

5.3.2主要是功能测试的内容，给定输入值，获得结果，并验证结果正确性。对于没有答案的对偶生成部分，我们采用人工评分的方法，打出三组分数，取均值获得生成对偶句的分数，从而判断功能完善程度。

### 测试项目与内容

1. 测试精确查询

精确查询共支持8中字符以及汉字的操作，对于其中的字符可以分为两类。

第一类是限制作者部分的字符，有‘&’与‘|’。

第二类是对于诗句的限制字符，有‘\*’,“[]”，“()”，“P”，“Z”，“a”。

所以，测试用例应需要准备的数量为112

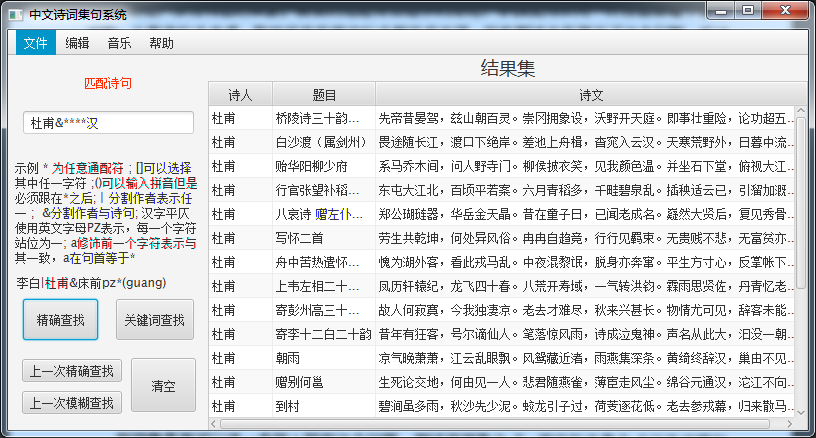
（“()”需要配合‘\*’使用所以不测试全括号情况）

但是，读取作者限制部分功能点代码位于一个方法，而诗句限制字符存在另一个方法。所以，实际准备样例测试主要使用情况，112中全部测试用例有很大一部分不符合实际使用情况，故不采用。例如：“PPPPP”，这样一个5言诗句全是平声字，不符合诗韵格律，虽然作为测试用例来讲没有任何问题，但是无效输入只会让搜索结果为0，不能检验方法功能正确性。所以”PZPPZ”这种输入更具有代表性,不仅符合格律而且可能有一批返回结果可供查验。

测试用例1

输入：杜甫&\*\*\*\*汉

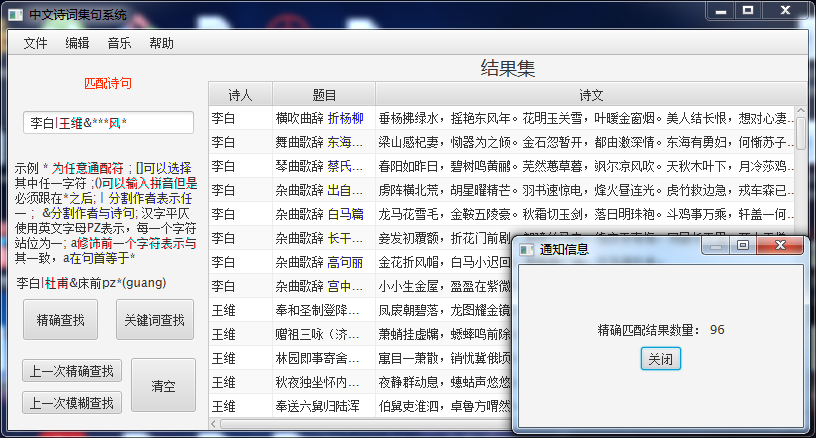
输出集：20首



测试用例2

输入：李白|王维&\*\*\*风\*

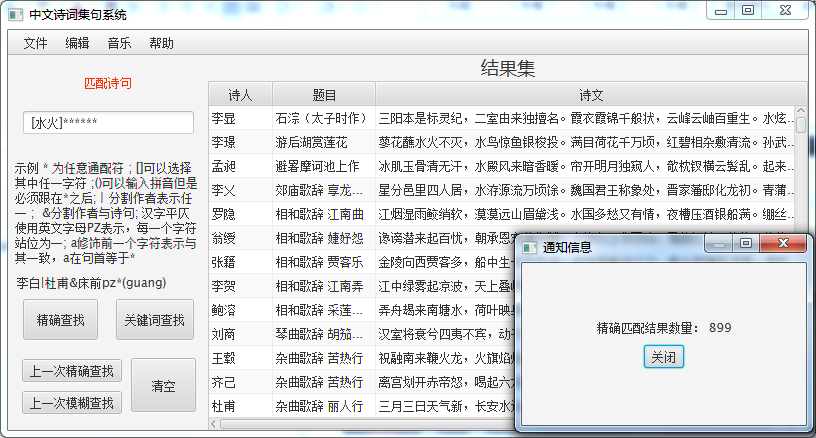
结果集：96首



测试用例3

输入：[水火]\*\*\*\*\*\*

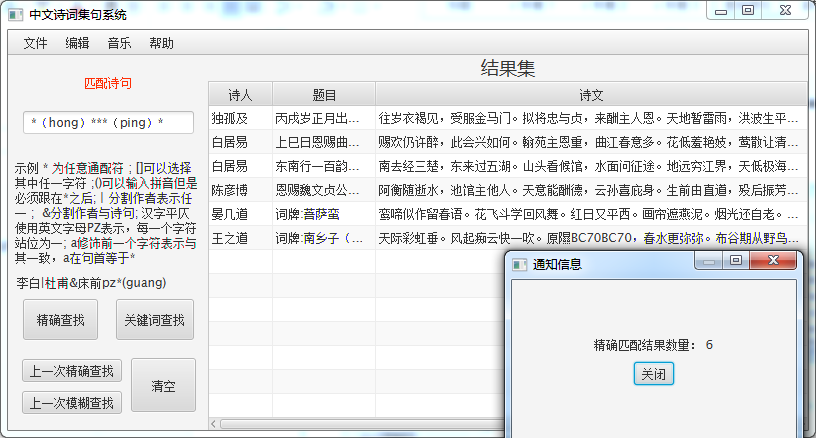
结果集：899首



测试用例4

输入：\*（hong）\*\*\*（ping）\*

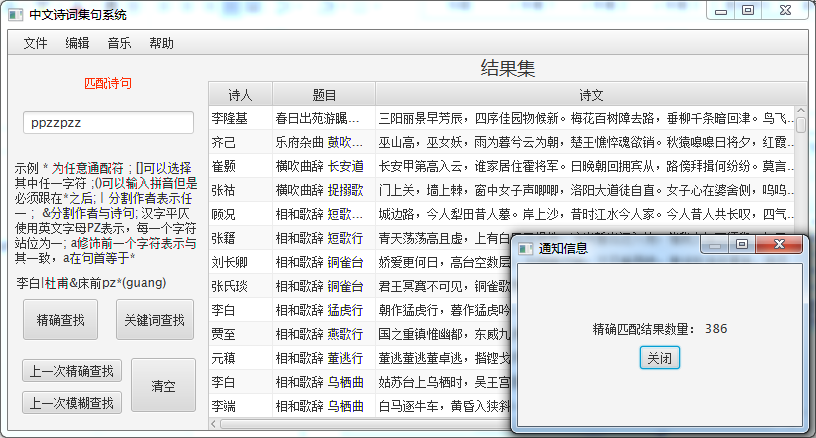
结果集：6首



测试用例5

输入：ppzzpzz

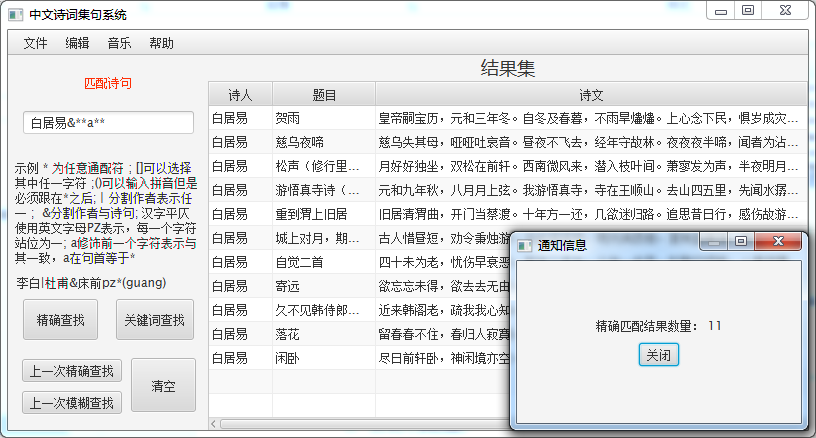
结果集：386首



测试用例6

输入：白居易&\*\*a\*\*

结果集：11首



至此，精确匹配的主要使用场景测试完毕，分别测试了带诗人、带多诗人、多字任选、平仄限制、拼音限制以及叠字查询。

1. 测试关键字查询

关键字查询，主要是测试在不提供词数的情况下，能否匹配多个不同长度的词语。

测试用例7

输入：百\*\*苦

结果集：4首

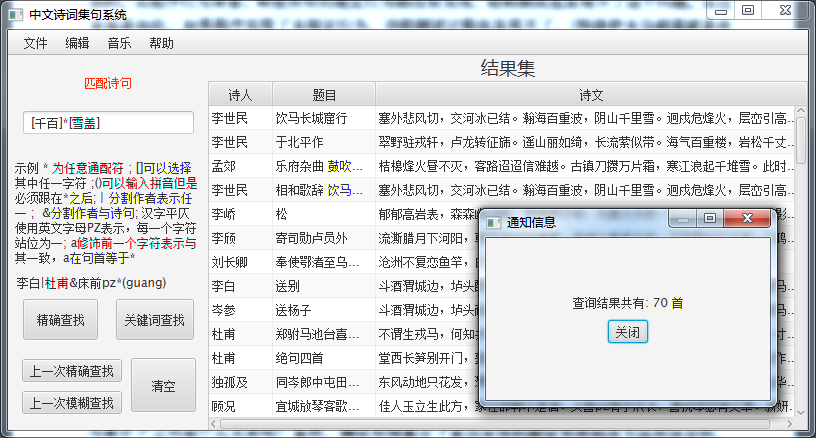


其中，“百炮资苦战”与“枉了百般辛苦”均成功匹配到“百\*\*苦”

测试用例8

输入：[千百]\*[雪盖]

结果集：70首



测试用例9

输入：娉婷

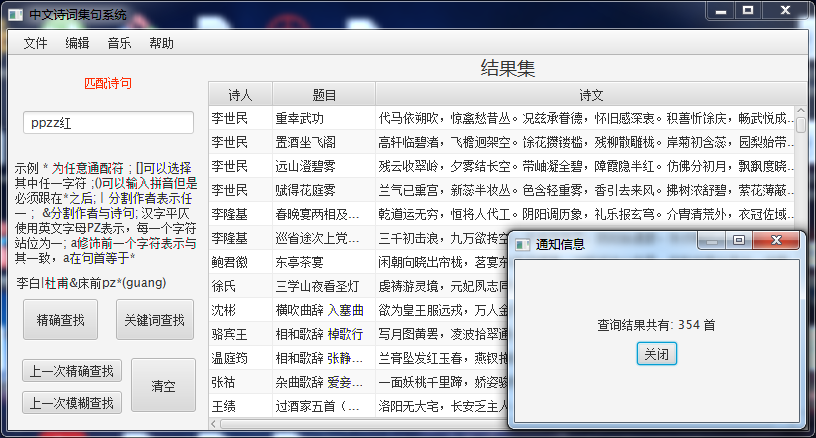
结果集：119首



测试用例10

输入：pzpp红

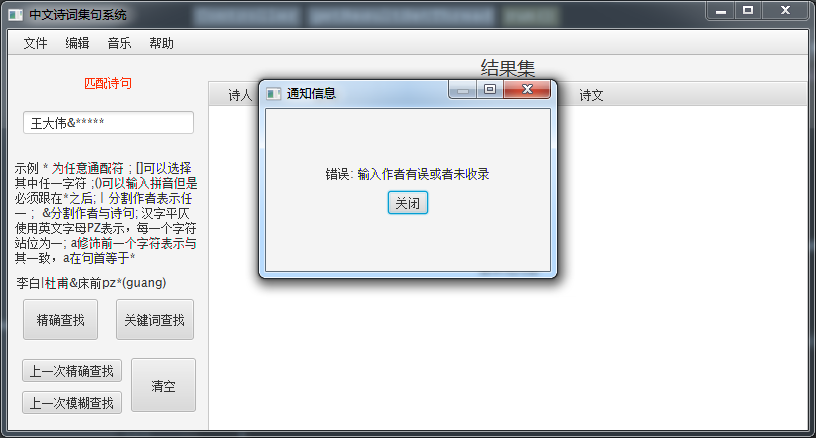
结果集：354首



测试用例11

输入：王大伟&\*\*\*\*\*

结果集：错误



错误日志

2017-05-11 22:37:20,658ERROR[com.Davi.poems.tools.analyze]-输入作者有误或者未收录

1. 测试对偶匹配

测试用例12

输入：春眠不觉晓

结果集：69首



测试用例13

输入：不如御风行万里

结果集：0首



1. 测试对偶生成

**第一组**

测试用例14

输入：不如御风行万里

使用分词方法：标准分词

结果集：制定5首



生成诗句有一定可读性，但是具体诗意不明显。

测试用例15

输入：月涌大荒流

使用分词：标准分词

结果集：指定5首



五言诗句明显可以有诗意，较七言表现更加

测试用例16

输入：不如御风行万里

使用分词：标准分词

结果集：指定5首



标准分词对仗并不工整

测试用例17

输入：不如御风行万里

使用分词：NLP分词

结果集：指定5首



NLP分词的对仗有明显进步

测试用例18

输入：不如御风行万里

使用分词：NShorPath分词

结果集：5首



发挥与标准分词类似，对仗并不工整

测试用例19

输入：不如御风行万里

使用分词：DijkstraShortPath分词

结果集：指定5首



表现与标准分词和NShortPath分词效果类似

综合表现，在输入句“不如御风行万里”，限制5首时，NLP分词表现最佳。

**第二组**

测试用例20

输入：春眠不觉晓

使用分词：标准分词

结果集：指定5首



其中，柳绿未经瞻一句接近对仗诗句，表现尚佳。

测试用例21

输入：春眠不觉晓

使用分词：NLP分词

结果集：指定5首



NLP分词表现好，5句对仗均有表现力

测试用例22

输入：春眠不觉晓

使用分词：NShortPath分词

结果集：指定5首



字形更为复杂，而含义却并不明确，表现不如前两种分词方法。

测试用例23

输入：春眠不觉晓

使用分词：DijkstraShortPath分词

结果集：指定5首



表现效果良好，接近标准分词结果。

结论，在输入诗句“春眠不觉晓”，指定生成5句时，NLP发挥依旧稳定，DijkstraShortPath与标准分词的结果也表现良好，只有NShortPath分词的对仗诗句表现最差。

在生成诗句的时间方面，每一单句诗句的生成时间从最短的8ms到1799ms均有分布，但是平均在100ms左右，下附部分时间数据日志。

2017-05-12 08:01:07,883INFO[com.Davi.poems.tools.analyze]-normalSegment生成对偶句所用时间 Time 23ms

2017-05-12 08:01:07,907INFO[com.Davi.poems.tools.analyze]-normalSegment生成对偶句所用时间 Time 24ms

2017-05-12 08:01:07,924INFO[com.Davi.poems.tools.analyze]-normalSegment生成对偶句所用时间 Time 17ms

2017-05-12 08:01:07,936INFO[com.Davi.poems.tools.analyze]-normalSegment生成对偶句所用时间 Time 12ms

2017-05-12 08:01:07,944INFO[com.Davi.poems.tools.analyze]-normalSegment生成对偶句所用时间 Time 8ms

2017-05-12 08:05:03,709INFO[com.Davi.poems.tools.analyze]-NLPSegment生成对偶句所用时间 Time 564ms

2017-05-12 08:05:05,508INFO[com.Davi.poems.tools.analyze]-NLPSegment生成对偶句所用时间 Time 1799ms

2017-05-12 08:05:06,162INFO[com.Davi.poems.tools.analyze]-NLPSegment生成对偶句所用时间 Time 654ms

2017-05-12 08:05:06,349INFO[com.Davi.poems.tools.analyze]-NLPSegment生成对偶句所用时间 Time 187ms

2017-05-12 08:05:06,460INFO[com.Davi.poems.tools.analyze]-NLPSegment生成对偶句所用时间 Time 111ms

2017-05-12 08:06:14,461INFO[com.Davi.poems.tools.analyze]-normalSegment生成对偶句所用时间 Time 43ms

2017-05-12 08:06:14,496INFO[com.Davi.poems.tools.analyze]-normalSegment生成对偶句所用时间 Time 34ms

2017-05-12 08:20:22,986INFO[com.Davi.poems.tools.analyze]-NshrotSegment生成对偶句所用时间 Time 163ms

2017-05-12 08:20:23,114INFO[com.Davi.poems.tools.analyze]-NshrotSegment生成对偶句所用时间 Time 127ms

2017-05-12 08:22:06,456INFO[com.Davi.poems.tools.analyze]-DjistraShrotSegment生成对偶句所用时间 Time 125ms

2017-05-12 08:22:06,558INFO[com.Davi.poems.tools.analyze]-DjistraShrotSegment生成对偶句所用时间 Time 102ms

2017-05-12 08:22:06,740INFO[com.Davi.poems.tools.analyze]-DjistraShrotSegment生成对偶句所用时间 Time 182ms

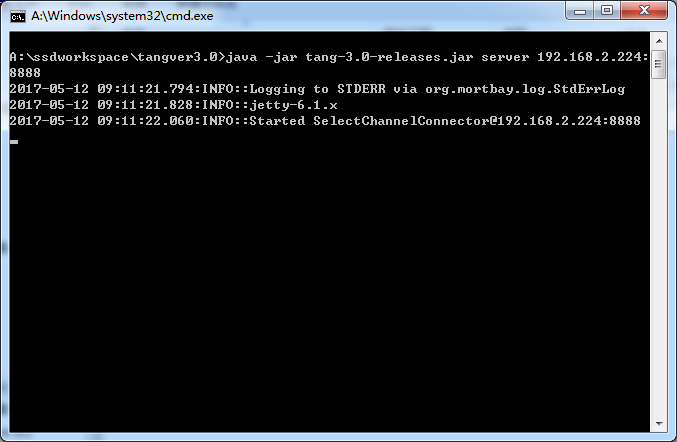
2017-05-12 08:22:06,886INFO[com.Davi.poems.tools.analyze]-DjistraShrotSegment生成对偶句所用时间 Time 145ms

2017-05-12 08:22:06,912INFO[com.Davi.poems.tools.analyze]-DjistraShrotSegment生成对偶句所用时间 Time 26ms

此外，随着字数的增加以及诗句数量的增加，对偶生成的所需时间也迅速增加，且每一次的生成时间也不相同。生成时间与句子的对仗工整程度并没有直接关联。

1. 非功能需求的实现测试

服务器模式启动



返回值

2017-05-12 09:14:03,120INFO[com.Davi.poems.net.myHttpClient]-

这首诗的题目是赠汪伦（白游泾县桃花潭，村人汪伦常酝美酒以待白）

这首诗的作者是李白

这首诗的内容是： 李白乘舟将欲行，忽闻岸上踏歌声。桃花潭水深千尺，不及汪伦送我情。

对偶句是：null

权重是0

这首诗的题目是山中问答

这首诗的作者是李白

这首诗的内容是： 问余何意栖碧山，笑而不答心自闲。桃花流水窅然去，别有天地非人间。

对偶句是：null

权重是0

这首诗的题目是东鲁门泛舟二首

这首诗的作者是李白

这首诗的内容是： 日落沙明天倒开，波摇石动水萦回。轻舟泛月寻溪转，疑是山阴雪后来。水作青龙盘石堤，桃花夹岸鲁门西。若教月下乘舟去，何啻风流到剡溪。

对偶句是：null

权重是0

这首诗的题目是曲江对酒

这首诗的作者是杜甫

这首诗的内容是： 苑外江头坐不归，水精春殿转霏微。桃花细逐杨花落，黄鸟时兼白鸟飞。纵饮久判人共弃，懒朝真与世相违。吏情更觉沧洲远，老大悲伤未拂衣。

对偶句是：null

权重是0

这首诗的题目是江畔独步寻花七绝句

这首诗的作者是杜甫

这首诗的内容是： 江上被花恼不彻，无处告诉只颠狂。走觅南邻爱酒伴，经旬出饮独空床。稠花乱蕊畏江滨，行步欹危实怕春。诗酒尚堪驱使在，未须料理白头人。江深竹静两三家，多事红花映白花。报答春光知有处，应须美酒送生涯。东望少城花满烟，百花高楼更可怜。谁能载酒开金盏，唤取佳人舞绣筵。黄师塔前江水东，春光懒困倚微风。桃花一簇开无主，可爱深红爱浅红。黄四娘家花满蹊，千朵万朵压枝低。留连戏蝶时时舞，自在娇莺恰恰啼。不是爱花即肯死，只恐花尽老相催。繁枝容易纷纷落，嫩叶商量细细开。

对偶句是：null

权重是0

这首诗的题目是昼梦

这首诗的作者是杜甫

这首诗的内容是： 二月饶睡昏昏然，不独夜短昼分眠。桃花气暖眼自醉，春渚日落梦相牵。故乡门巷荆棘底，中原君臣豺虎边。安得务农息战斗，普天无吏横索钱。

对偶句是：null

权重是0

2017-05-12 09:14:03,121INFO[com.Davi.poems.net.myHttpClient]-返回 200

服务器端日志

2017-05-12 09:14:02,673INFO[com.Davi.poems.net.handler]-李白|杜甫&桃花\*\*\*\*\*

2017-05-12 09:14:02,778INFO[com.Davi.poems.net.handler]-result size: 6

1. 测试背景音乐

可以完整播放，在查询时可以持续播放。

1. 程序查询时状态

程序查询时有卡顿，期间不能进行其他操作，无法关停音乐。

### 测试的结论

首先，软件已经完成了需求说明中的大部分方法，但是在对偶生成的诗句的工整程度上，并没有达到原诗句等级。而诗句查询方面，可以较完整的实现功能，而且查询流畅度尚佳，但是查询时系统进入无法反应状态，必须等待结果出现之后界面才会刷新。对偶句匹配因为数据原因，无法达到对于所有汉字的精确对偶匹配从结果来看结尾可惜，但改进字典库是其中一个较为关键的方向。而在对偶生成部分，结合绪论中关于研究现状的表述，在机器自动生成诗句的方向下，仅仅是根据已有的训练集在加上简单的对偶模型并不能达到完美对仗的水平。而且，在服从概率分布的随机生成汉字的算法帮助下，最好的表现情况出现在NLP分词对仗较为工整，但也接近该算法模型的极限。对偶生成的改进方向，应该从现有框架中跳出，使用新的技术才可以有革命性的进步，更加近似于人的作诗风格。根据目前的研究成果，RNN（卷积神经网络）较为适合这种对于语言层面的学习，但限于本人水平有限、没有训练集群以及神经网络研究经验，不能再本片论文中继续增加RNN模型对于对偶生成的改进。

# 结论

中文集句系统，是以搜索、处理诗句之中的汉字为核心理念，提供诗句搜索，诗句匹配，关键字匹配、对偶句匹配、对偶句生成等功能的软件系统。经过较完善的软件工程模型—瀑布模型的流程监督，有完整的需求分析、概要设计、详细设计、软件实现、软件测试的流程。整体系统分为3个模块，用户交互的UI模块、数据查询的逻辑作用模块以及数据载入模块，使用Java作为程序开发语言，运用JavaFX解决用户交互界面设计，采用XML作为源数据的组织形式，有客户端模式以及服务器模式两种状态，分别提供界面的功能展示以及HTTP的网络请求返回功能。在检索部分，自定义了诗句的规则描述符号，使用符号组合进行诗句查询，在对偶生成方面，运用成熟的中文分词开源包，对诗句中的中文进行分词从而获取词性信息，再加上字典上的平仄从而生成对偶词，并且在生成的字词的概率参考了数据库中诗文的汉字出现概率，提高了结果的可读性。

程序开发参考了指导教师高晓程老师的原型Demo程序，根据本科阶段的开发经验，使用Maven管理Java的依赖，使用Git管理程序版本，使用log4j打印程序日志，部分的功能模块，如音乐播放以及服务器模式采用了线程技术，努力的使程序的运行更加流畅且效率提高。

本系统在帮助中文诗句爱好学习诗句、联系平仄等方面有积极的帮助作用，在闲暇时查询古诗不失为放松心情、排解压力的不二选择。软件系统在诗句的搜索、数据集的拓展、字典的完善以及对偶句生成算法上依然具有广阔的进步空间，是未来改进的主要方向。

# 致谢

至此，本篇论文即将结束。在漫长的本科学习阶段，感谢我的导师高晓程，不仅在操作系统课程中为我答疑解惑，更是在最后的毕业设计为我提供了这个有趣而富有挑战性的课题，感谢我的导员在我的学习生活中无私的帮助，感谢我的任课教师，不仅教会了我的专业知识，更教会了我学习的方法。学海无涯，虽然大学的学习即将结束，但是在计算机领域，时时刻刻都在创新，在诞生新的方式方法，毕业是新的学习开始。在毕业设计的制作方面，我遇到过瓶颈，曾经彻夜DEBUG，测试程序方法的错误与漏洞，曾为了课设而废寝忘食，最终在规定的时间内完成了任务。最后的结果无论如何，我认为已经对得起自己几个月来的努力，成果无论是巨大还是微小，都为自己本科生涯画上了一个句号。在以后的工作学习中，要戒骄戒躁，将本次的经验作为财富，努力的前进。

最后，感谢在我本科学习生活的过程中给予我指导帮助的老师、同学们。还要感谢坚持到最后的自己，谢谢。

# 参考文献

1. hankcs HanLP自然语言处理包开源 <http://www.hankcs.com/nlp/hanlp.html> 2015.03
2. 王力 诗词格律概要 北京：北京出版社，1979
3. Li Zhongguo, Maosong Sun. Punctuation as implicit annotations for chinese word segmentation[J]. Computational Linguistics, 2009 35(4):505-512.
4. Orcale JavaSE 8 Document about JavaFX <http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784> 2013.09
5. XMan JavaFX中文、OSGI、Eclipse开源资料 <http://www.javafxchina.net/blog/docs> 2017
6. Oracle JavaFX Document <http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#BABEDDGH> 2013.09
7. hankcs 词图的生成 <http://www.hankcs.com/nlp/segment/the-word-graph-is-generated.html> 2014.05
8. hankcs 词性标注 <http://www.hankcs.com/nlp/part-of-speech-tagging.html#h2-8> 2014.11
9. 刘河生 高小榕 杨福生. 隐马尔可夫模型的原理与实现[J]. 国际生物医学工程杂志, 2002 25(6):253-259.
10. Bilmes J A. What HMMs Can Do[M]. Oxford University Press, 2006.
11. hankcs 通用维比特的Java实现 <http://www.hankcs.com/nlp/general-java-implementation-of-the-viterbi-algorithm.html> 2014.09
12. 刘向威 NLP技术在中文信息检索中的应用研究[D] 天津大学, 2005.
13. Allenxl NLP自然语法解析 <http://blog.csdn.net/lanxu_yy/article/details/37700841，2014.07>
14. hankcs N最短路径的Java应用与实现 <http://www.hankcs.com/nlp/segment/n-shortest-path-to-the-java-implementation-and-application-segmentation.html>
15. Thomas H.Cormen Charles E.Leiserson Ronald L.Rivest Clifford Stein. 算法导论 北京：机械工业出版社

# 附录1 英文论文原文

**What HMMs Can Do**

Jeff Bilmes

bilmes@ee.washington.edu

Dept of EE, University of Washington

Seattle WA, 98195-2500

University of Washington, Dept. of EE, UWEETR-2002-0003

January 2002

**Abstract**

Since their inception over thirty years ago, hidden Markov models (HMMs) have have become the predominant methodology for automatic speech recognition (ASR) systems — today, most state-of-the-art speech systems are HMM-based. There have been a number of ways to explain HMMs and to list their capabilities, each of these ways having both advantages and disadvantages. In an effort to better understand what HMMs can do, this tutorial analyzes HMMs by exploring a novel way in which an HMM can be defined, namely in terms of random variables and conditional independence assumptions. We prefer this definition as it allows us to reason more throughly about the capabilities of HMMs. In particular, it is possible to deduce that there are, in theory at least, no theoretical limitations to the class of probability distributions representable by HMMs. This paper concludes that, in search of a model

to supersede the HMM for ASR, we should rather than trying to correct for HMM limitations in the general case, new models should be found based on their potential for better parsimony, computational requirements, and noise insensitivity.

**Introduction**

By and large, automatic speech recognition (ASR) has been approached using statistical pattern classification [29, 24, 36], mathematical methodology readily available in 1968, and summarized as follows: given data presumably representing an unknown speech signal, a statistical model of one possible spoken utterance (out of a potentially very large set) is chosen that most probably explains this data. This requires, for each possible speech utterance, a model governing the set of likely acoustic conditions that could realize each utterance.

More than any other statistical technique, the Hidden Markov model (HMM) has been most successfully applied to the ASR problem. There have been many HMM tutorials [69, 18, 53]. In the widely read and now classic paper [86], an HMM is introduced as a collection of urns each containing a different proportion of colored balls. Sampling (generating data) from an HMM occurs by choosing a new urn based on only the previously chosen urn, and then choosing with replacement a ball from this new urn. The sequence of urn choices are not made public (and are said to be “hidden”) but the ball choices are known (and are said to be “observed”). Along this line of reasoning, an HMM can be defined in such a generative way, where one first generates a sequence of hidden (urn) choices, and then generates a sequence of observed (ball) choices.

For statistical speech recognition, one is not only worried in how HMMs generate data, but also, and more importantly, in an HMMs distributions over observations, and how those distributions for different utterances compare with each other. An alternative view of HMMs, therefore and as presented in this paper, can provide additional insight into what the capabilities of HMMs are, both in how they generate data and in how they might recognize and distinguish between patterns.

This paper therefore provides an up-to-date HMM tutorial. It gives a precise HMM definition, where an HMM is defined as a variable-size collection of random variables with an appropriate set of conditional independence properties. In an effort to better understand what HMMs can do, this paper also considers a list of properties, and discusses how they each might or might not apply to an HMM. In particular, it will be argued that, at least within the paradigm offered by statistical pattern classification [29, 36], there is no general theoretical limit to HMMs given enough hidden states, rich enough observation distributions, sufficient training data, adequate computation, and appropriate training algorithms. Instead, only a particular individual HMM used in a speech recognition system might be inadequate. This perhaps provides a reason for the continual speech-recognition accuracy improvements we have seen with HMM-based systems, and for the difficulty there has been in producing a model to supersede HMMs.

This paper does not argue, however, that HMMs should be the final technology for speech recognition. On the contrary, a main hope of this paper is to offer a better understanding of what HMMs can do, and consequently, a better understanding of their limitations so they may ultimately be abandoned in favor of a superior model. Indeed, HMMs are extremely flexible and might remain the preferred ASR method for quite some time. For speech recognition research, however, a main thrust should be searching for inherently more parsimonious models, ones that incorporate only the distinct properties of speech utterances relative to competing speech utterances. This later property is termed structural discriminability [8], and refers to a generative model’s inherent inability to represent the properties of data common to every class, even when trained using a maximum likelihood parameter estimation procedure. This means that even if a generative model only poorly represents speech, leading to low probability scores, it may still properly classify different speech utterances. These models are to be called discriminative generative models.

Section 2 reviews random variables, conditional independence, and graphical models (Section 2.1), stochastic processes (Section 2.2), and discrete-time Markov chains (Section 2.3). Section 3 provides a formal definition of an HMM, that has both a generative and an “acceptive” point of view. Section 4 compiles a list of properties, and discusses how they might or might not apply to HMMs. Section 5 derives conditions for HMM accuracy in a KullbackLeibler distance sense, proving a lower bound on the necessary number of hidden states. The section derives sufficient conditions as well. Section 6 reviews several alternatives to HMMs, and concludes by presenting an intuitive criterion one might use when researching HMM alternatives

# 附录2 译文

**隐马尔可夫模型可以做什么**

杰夫·比尔莫斯

华盛顿大学电子信息学院

Seattle WA, 98195-2500

University of Washington, Dept. of EE, UWEETR-2002-0003

January 2002

**摘要**

经过了30多年的研究，隐马尔可夫模型(HMMs)已经变成了构造自动语言识别(ASR)系统的占主导地位的方法论。今天，目前最先进的语言系统都以隐马尔可夫模型作为基础。我有很多种方法来介绍隐马模型并且列出他们的能力。但每一种解释利弊都兼而有之。为了更好地理解隐马模型可以做什么这一问题，本篇论文将探索一个隐马模型的定义新方法，即根据随机变量和条件独立假设进行定义。我们更加推崇这种定义方式，因为我们可以更加深入了解一个隐马模型的能力。尤其是这种方法可以在理论层面上推断没有理论限制的概率分布隐马尔可夫模型。论文的结论是为了需找一个模型代替自动语言识别系统中的的隐马尔可夫模型，我们不应该局限的对隐马模型进行参数修正，而要设计一个更加节约计算资源、对噪声更不敏感的模型。

**简介**

总体来说，自动语言识别(ASR)技术使用统计模式分类[29,24,36],所需要的数学理论基础在1968年已经完善可用。总结如下：对于给定的代表位置语言的信号数据，根据一个（非常巨大的权威）可能性语言表达统计模型来挑选出最可能解释一段文字的语言表达。这就要求，对于可能的语言表达模型可以调节近似的语言表达环境集合。

比起任何其他的统计技术，Hidden Markov模型(HMM)已经最佳的应用于ASR问题。目前有很多的HMM教程[69,18,53]。经过深入的阅读现在的经典HMM论文[86]，一个HMM模型可以形象的表述为一个含有不同彩色球比例的壶，从一个HMM模型中抽样(生成数据)，仅根据之前选择的壶来选择一个新的壶来代替原来的壶。壶的选择顺序不是公开的(称为“隐藏”)，但是球的选择却是已知的(称为“观察”)。沿着这样的逻辑推理，HMM可以被定义成一种生成的方式，首先生成一个隐藏选择壶的序列，然后生成一个观察球的序列。

对于统计语言识别来讲，不仅要关注HMM模型产生的数据，而且更加重要的是，关注HMM模型的分布情况，以及这些分布在不同的语言表达相互的区别。因此本文展现从另外一个观点观察HMMs模型，可以提供额外观察HMM模型的角度，无论是在模型生成数据还是识别和区分模式。

因此本文提供了一种最新的HMM教程。论文给出了，HMM的精确定义，HMM是一个可变化的集合以及一组适当的独立属性条件随机变量。为了更好的了解一种HMM可以做什么，本文额外考虑了属性列表，以及属性适用以及不适用的情况。在这里有争论，至少分类统计模式[29,36]提供了范式，但是没有一般的理论限制HMMs拥有足够的隐藏状态、足够的分类观察、充足的训练数据、充足的计算量以及合适的训练算法。相反，只使用一个特定的HMM模型的语言识别系统可能是不足够的。这似乎解释了连续语音识别精度改善的困难，以及制作取代HMMs模型的困难。

本论文不讨论HMMs是否应该作为ASR的最终理论模型。相反，本文希望提供对于HMMs可以做什么的更加深入的理解，从而更加清晰的了解HMM的局限性。所以，HMM可能会在更加优越的模型出现后被放弃。确实，隐马模型十分灵活，可能在很长一段时间依然是ASR的首选算法。然而，对于语音识别的研究，一个重要的动力，是寻找更加简洁的模型，只包含与竞争语言表述不同的清楚的语言表述特点。这个之后出现的属性被称为结构特征[8]，是指一个生成模型的固有的表示数据共同特性的各种方面，训练时甚至使用最大似然估计参数来估算。这意味着，生成模型即使仅代表语音，获得低概率的分数，但是仍然可以正确分类不同语言表述。这些模型被称为判断生成模型。

第2章回顾随机变量，有条件见的独立性和图形化模型（第2.1节），随机过程（第2.2节），离散时间的马尔可夫链（第2.3节）。第3章，提供HMM的一个形式化的定义，同时从生成性以及“接受”的角度来研究。第4章，属性的收集，并探讨其中可以或者不可以使用的隐马尔可夫模型。第5章介绍在一个叫做KullbackLeibler距离精度条件下，试验在较低的限制条件下，必要的隐藏层数量。第6章，回顾HMM模型的集中不同的模型，并且提出一个可以直接使用在研究HMM替代模型上的标准的结论。

1. 韵 平声共30韵，上声29韵，去声30韵，入声17韵

   上平声 一东 二冬 三江 四支 五微 六鱼 七虞 八齐 九佳 十灰 十一真 十二文 十三元 十四寒 十五删

   下平声 一先 二萧 三肴 四豪 五歌 六麻 七阳 八庚 九青 十蒸 十一尤 十二侵 十三覃 十四盐 十五咸

   上声 一董 二肿 三讲 四纸 五尾 六语 七麌 八荠 九蟹 十贿 十一轸 十二吻 十三阮 十四旱 十五潸 十六铣 十七筱 十八巧 十九皓 二十哿 二十一马 二十二养 二十三梗 二十四迥 二十五有 二十六寝 二十七感 二十八琰 二十九豏

   去声 一送 二宋 三绛 四寘 五未 六御 七遇 八霁 九泰 十卦 十一队 十二震 十三问 十四愿 十五翰 十六谏 十七霰 十八啸 十九效 二十号 二十一个 二十二祃 二十三漾 二十四敬 二十五径 二十六宥 二十七沁 二十八勘 二十九艳 三十陷

   入声 一屋 二沃 三觉 四质 五物 六月 七曷 八黠 九屑 十药 十一陌 十二锡 十三职 十四缉 十五合 十六叶 十七洽 [↑](#footnote-ref-1)