

第十二届中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

第十二届“认证杯”数学中国

数学建模网络挑战赛

承 诺 书

我们仔细阅读了第十二届“认证杯”数学中国数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们接受相应处理结果。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

我们的参赛队号为：**1142**

参赛队员（签名）：

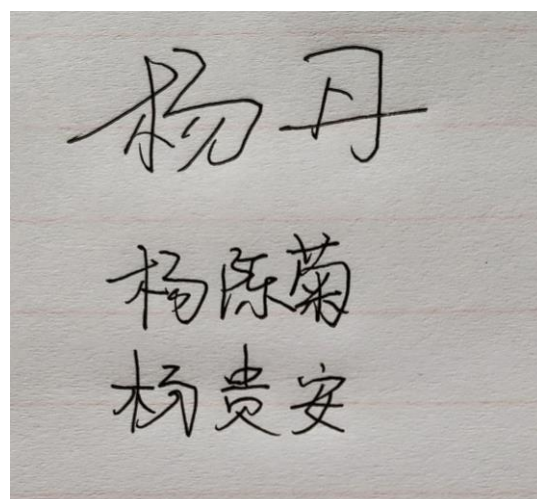
队员 1：

队员 2：

队员 3：

参赛队教练员（签名）：

参赛队伍组别（例如本科组）：研究生组



杨丹
杨陈菊
杨贵安

第十二届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

第十二届“认证杯”数学中国

数学建模网络挑战赛

编号专用页

参赛队伍的参赛队号：（请各个参赛队提前填写好）：
1142

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

第十二届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

2019 年第十二届“认证杯”数学中国 数学建模网络挑战赛第二阶段论文

题 目 基于统计和迭代匹配的未知语言文本片段提取模型

关 键 词 统计 MC 减法聚类 迭代匹配搜索 汉明距离 Gauss

摘 要：

任何自然语言系统都可看作是表达意义的符号系统，本文提出的模型的目标是如何有效且快速地从完全未知的语言文本中提取出可能具有某个固定含义的片段，并考虑了在记录文本过程中由于技术限制出现的错误，能够最大限度提取出可能由于记录错误导致出错的片段，并分析错误的类型（替换错误、删失错误、插入错误）。

针对问题一，本文假设希望获取到的片段长度固定的为 15 个字母，从时间上和空间上都简化了复杂度。

首先我们按照要求模拟了 30 段长度在 5000~8000 个字母之间且含有错误的外星文本。具体做法是先构造一个每个字母呈 Gauss 分布的片段库，每个片段长度固定为 15 个字母；再根据片段库里的片段构造出 30 段正确的文本，每个片段在文中也是呈 Gauss 分布；接着又根据正确的文本构造出 30 段错误的文本，其特点是每个字母有 1/5 的出错概率，并且每个错误类型出现的概率皆为 1/3。

然后，本文对外星文本进行大量统计和分析，寻找到截取出的每个片段的三个属性特征，由此建立三维坐标系。具体做法是首先以窗体形式截取出所有片段，固定窗体宽度为 15；然后基于马尔可夫（Markov）模型获取片段的概率；紧接着回到外星文本统计出每个字母的频率，观察其分布；最后统计出每个片段中不同符号（字母）的个数，作为片段的第三个特征量；经过这些工作，就可以建立三维坐标系。

接着，对于上一步所得到的片段三维特征量进行减法聚类（subtrative clustering method, SCM）便得出中心点的三维坐标，根据中心点的坐标确定出中心片段，即我们的目标片段。最后将取得的目标片段与片段库 1 作比对，正确率为 73.3%。

最后还对模型一做了扩展，在模型一的基础上，结合汉明距离(Hamming Distance)分析出错误片段的错误类型。分析结果表明错误片段的错误类型基本可以确定。

针对问题二，本文通过改进模型一，建立了模型二，改进内容最主要有以下四个方面：

第一方面，将模型一的片段库里的定长片段改为长度在 15~21 字母之间的不定长片段，其中 15~21 这一片段长度是根据第一阶段的问题而假定的，在模型二中只是用来说明模型，长度的范围可以灵活改变。

第二方面是建模的难点，不定长的片段应如何截取才能得到我们所需要的片段，我们丢弃了传统的方法（传统方法是将每个长度的片段都截取一遍），而本文通过对片段进行仔细研究找到这样一个特点：从文本的同一位置出发，非最短长度（这里指长度为 16、17、18、19、20、21 的片段）必定已经包含最短长度的片段（这里指长度为 15 的

第十二届中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

片段)，即最短长度的片段一定是非最短长度的片段的子串，此处忽略非最短片段间的相互包含关系（如：长度为 19 的片段是长度为 21 片段的子片段）。根据这一特点，本文只选取最短片段长度作为窗体大小截取子串，再根据后面的算法找出子串的“后半段”来确定目标片段。

第三方面是建模的重点，前面只截取了片段的子串，那么如何找出连接在子串后的后半段是该步骤要解决的问题。解决方法是迭代匹配搜索算法，把中心子串带回外星文本按一定的概率标准和长度限制进行“首字母位置不变，尾字母右移 1 位”匹配搜索，直至确定出目标片段。最后将取得的目标片段与片段库 1 作比对，正确率为 83.3%。

第四个方面，分析错误类型时，长短不一的两个片段先使用“-”进行补位后再进行汉明距离的计算。

另外本文运用 Java、Matlab 语言，用 Excel 工具做辅助，对模型和问题进行了仿真实验，验证了模型的可行性。

最后本文还对模型进行了量化评价，总体来说通过对问题一的分析基本建立起了整体模型，通过问题二对模型一进行了改进，模型二的可靠性较强，有效性适中，但模型的使用灵活性大，范围广，还可用于自然语言处理（Natural Language Processing, NLP）、机器学习(Machine Learning)等方面，是一个应用性能比较强的模型。

参赛队号：1142

所选题目：B 题

参赛密码
(由组委会填写)

第十二届中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

英文摘要

Abstract Any natural language system can be regarded as a symbolic system for expressing meaning. The goal of the model in this paper is that effectively and quickly extract fragments from a completely unknown language text that possibly have a certain meaning. The model can maximum extract errors of fragments that may be caused by limitation of recording technology, and analyze the type of errors that includes replacement errors, censored errors and insertion errors.

For the first problem, this paper assumes that the length of the fragment that you want to acquire is fixed at 15 letters, which simplifies the complexity both in time and space.

Firstly, we simulated text of 30 segments that is lengths between 5000 and 8000 letters and contain errors according to the requirements. The specific method is to construct a fragment that obey Gauss distribution for each letter. Each fragment is fixed to 15 letters in length. 30 pieces of correct text are constructed according to the fragments in the fragment library, and each fragment is also Gaussian in the text. According to the correct text, 30 pieces of erroneous text are constructed, which is characterized by a $1/5$ error probability for each letter, and each error type has a probability of $1/3$.

Secondly, the paper conducts a large number of statistics and analysis on the extraterrestrial text, and finds three features of each segment. Thereby, establishing a three-dimensional coordinate system. The specific method is to extract firstly all the fragments from text, and fix the width of the form to 15. Obtaining the probability of the fragment that based on the Markov model. Calculating the extraterrestrial text to count the frequency of each letter. The number of different letters in each segment is counted as the third feature quantity of the segment. After these works, a three-dimensional coordinate system can be established.

Thirdly, the subtrative clustering method (SCM) of the three-dimensional feature quantity of the segment obtained in the previous step is used to obtain the three-dimensional coordinates of the center point, and the center segment is determined according to the coordinates of the center point, that is, our target segment. Finally, the target segment obtained is compared with the segment library 1, and the correct rate is 73.3%.

Finally, the model is also extended. Based on the model one, the Hamming Distance is used to analyze the error type of the error segment. The analysis results show that the error type of the error segment can be basically determined.

For the second problem, this paper establishes the second model by improving the model one. The main content of the improvement is as follows:

第十二届中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

In the first, the fixed length segment in the fragment library of model one is changed to an indefinite length segment between 15 and 21 letters, wherein the length of the segment 15-21 is assumed according to the problem of the first phase. Model 2 is only used to illustrate the model, and the range of lengths can be flexibly changed.

Secondly, how to cut the fragment of indefinite length to get the fragment we need, we discard the traditional method (the traditional method is to intercept each length of the fragment), and this article passes the fragment. Careful study to find such a feature: from the same position of the text, non-minimum length (here, the length of 16, 17, 18, 19, 20, 21) must have the shortest length of the segment (here the length is 15). The fragment of the shortest length must be a substring of the fragment of the shortest length, and the mutual inclusion relationship between the non-shortest fragments is ignored here (for example, a fragment of length 19 is a sub-segment of length 21). According to this feature, this paper only selects the shortest segment length as the form size to intercept the substring, and then finds the target segment by finding the "second half" of the substring according to the following algorithm.

Thirdly, in the previous section, only the substrings of the fragments were intercepted. How to find the second half after the substrings is the problem to be solved in this step. The solution is to iteratively match the search algorithm, and bring the center substring back to the extraterrestrial text according to a certain probability criterion and length limit, and the "first letter position is unchanged, the tail letter is shifted to the right by 1 bit" to match the search until the target segment is determined. Finally, the target segment obtained is compared with the segment library 2, and the correct rate is 83.3%.

Finally, when analyzing the type of error, the two segments of different lengths are first filled with '-' and then the Hamming distance is calculated.

In addition, this paper uses Java and Matlab language, assists with the Excel tool, and carries out simulation experiments on the model and the problem to verify the feasibility of the model.

Finally, the model is also quantitatively evaluated. In general, the overall model is basically established through the analysis of problem 1. The problem is improved by the second problem. The reliability of model 2 is strong and the effectiveness is moderate, but the model It has a wide range of flexibility and can be used in natural language processing (NLP), machine learning (Machine Learning), etc. It is a model with strong application performance.

目录

一、研究背景.....	3
二、问题重述与建模思路.....	3
2.1 问题重述.....	3
2.2 建模思路.....	3
2.2.1 针对问题一.....	4
2.2.2 针对问题二.....	4
三、模型假设及符号说明.....	4
3.1 模型假设.....	4
3.2 名词定义.....	4
3.3 符号说明.....	5
四、问题一建模求解.....	6
4.1 模型一流程.....	6
4.2 模拟外星文本.....	7
4.2.1 建立片段库 1.....	7
4.2.2 生成正确文本.....	7
4.2.3 生成错误文本.....	7
4.3 基于统计获取全部固定长度片段.....	8
4.3.1 片段截取.....	8
4.3.2 基于马尔可夫链的初步统计.....	9
4.3.3 计算字母频率.....	10
4.3.4 确定三维坐标.....	10
4.4 减法聚类.....	10
4.5 基于中心片段和汉明距离分析错误类型.....	12
4.5.1 中心片段与非中心片段汉明距离.....	12
4.5.2 错误类型分析.....	13
4.5.3 聚类结果相似度验证.....	15
五、问题二建模求解.....	16
5.1 模型二流程.....	16
5.2 模拟外星文本.....	16
5.3 基于统计获取全部片段.....	17
5.3.1 片段截取.....	17
5.3.2 基于统计确定三维坐标.....	17

5.4 减法聚类	17
5.5 迭代匹配搜索目标片段	18
5.6 基于目标片段和汉明距离分析错误类型	20
5.6.1 目标片段与非中心子串汉明距离	20
5.6.2 错误类型分析	20
六、模型的评价与推广	22
6.1 模型性能分析	22
6.1.1 从统计的角度分析	22
6.1.2 从时间复杂性和空间复杂性角度分析	22
6.2 模型的优点	23
6.3 模型的缺点	23
6.4 模型推广	23
参考文献	24
附录一 程序	25
附录 1.1 问题一程序	25
附录 1.2 问题二程序	36
附录二 运行结果（部分）	42
附录 2.1 问题一结果	42
附录 2.2 问题二结果	45

一、研究背景

浩瀚无垠的宇宙中存在着众多星球，地球文明置身其中无异于沧海一粟，对宇宙文明的探究逐渐成为地球文明的一大重要“课题”。2018 年 10 月澳大利亚 CSIRO 射电望远镜探测到 19 次神秘射电波，最近的 FRB 大约距离地球 4.25 亿光年，被命名为 FRB 171020。这组爆发几乎是已知 FRB 总数的两倍，仅仅持续几毫秒，但产生这些强大爆发的原因仍然是个谜团。有专家认为爆发可能来自中子星，而 NASA 强调这些信号很有可能是外星人信息。2019 年北京时间 5 月 17 日，据宇宙杂志网站报道，目前研究人员表示外星人可能会在黑洞附近使用引力波进行通讯，干涉仪参数的微小变化表明地球并不孤单，宇宙中或许存在着其他高智慧生命形式。有物理学家小组建议欧洲航天局拟定 2034 年发射的“激光干涉仪太空天线（LISA）”可作为高等外星文明发送信号的探测器。若真的有外星文明存在，又该如何与之交流？早在 17 世纪初，意大利哲学家和天文学家伽利略就认为，数学语言是解读宇宙语言的钥匙。人类探索宇宙的脚步从未停止过。

物理学家霍金生前发出最后警告：千万不要联系或接触外星文明！他的警告与刘慈欣在《三体》中提出的“黑暗森林”法则不谋而和，而解读外星语言必然是探析外星文明意图的第一步。假若在未来的某一天，地球收到了来自外太空高智慧文明的信号，对外星文明信号的解密工作由此秘密展开……

二、问题重述与建模思路

2.1 问题重述

现有由 20 个字母构成的未知语言，没有标点，没有空格。假设我们已经获取了 30 段由该语言写成的文本，我们无法理解其规律及含义，希望对这种语言开展研究。有一种思路是设法在不同段文本中搜索共同出现的字母序列的片段。语言学家猜测：如果有的序列片段在每段文本中都会出现，这些片段就很可能具备某种固定的含义（类似词汇或词根），我们以此入手进行进一步的研究。在文本的获取过程中，由于人类对记录技术的限制，可能有一些位置出现了记录错误。可能的错误分为如下三种：

1. 删失错误：丢失了某个字母；
2. 插入错误：新增了原本不存在的字母；
3. 替换错误：某个字母被篡改成了其他的字母。

现假设我们已经获取了 30 段文本，每段文本的长度都在 5000 - 8000 个字母之间。我们希望找到的片段的长度为 15 个字母。由于技术的限制，当我们在记录每个字母时，都可能有五分之一的概率发生错误。错误类型可能为删失错误、插入错误或替换错误，每个错误只涉及一个字母，且每个错误的发生是独立的。请你设计合理的数学模型，快速而尽可能多地找到符合要求的片段，并自行编撰算例来验证算法的效果。如果我们事先不知道所寻找的片段的长度，算法应如何改进。

2.2 建模思路

解读未知文本的过程，第一步是寻找很可能具备某种固定的含义（类似词汇或词根）的片段。这些希望被找到的片段出现频率不一，从经验的角度出发，文本的基元出现频率一般服从 Gauss 分布，为此，我们假设各片段、各字母在文本中出现的频率大致服从 Gauss 分布。

2.2.1 针对问题一

问题一的要求是：有 30 段长度为 5000~8000 的外星文本，片段长度固定为 15，每个字母有 20% 的概率发生错误，考虑有三种错误（删失错误、插入错误、替换错误），每个错误只涉及一个字母，且错误是独立发生的。要求快而尽可能多地找到符合要求的片段。正对该问题本文的基本思路是首先构造外星文本，对文本截取片段、做一些基础统计，由此建立三维坐标系，然后通过减法聚类找到目标片段，到此，我们还要根据聚类中心点以及汉明距离分析出有可能出现错误的片段和错误类型。

2.2.2 针对问题二

问题二的要求是：在问题一的基础上假设事先不知道所寻找的片段的长度，要如何改进模型一。本文假设片段的长度在 15 - 21 个字母之间（参考第一阶段的问题要求），设想在截取片段的过程中，在同一个位置往后截取长度为 15 的片段 S_1 和截取长度为 18（可以为 16、17、18、19、20、21）的片段 S_2 ，那么 S_2 必定包含 S_1 ，换句话说， S_1 是 S_2 的子串。在模型一的基础上我们改进的部分是：

- (1) 建立长度不固定的片段库；
- (2) 先只截取最小字串长度（15）的所有片段
- (3) 减法聚类的结果其实就是要找的目标片段的子串，把子串带入外星文本搜索以这个子串为字串的更长片段，根据频率确定出本来该有的片段长度

三、模型假设及符号说明

3.1 模型假设

在求解过程中，假设：

- (1) 文本中一共由 20 个不同的符号构成，符号的形状并不是主要研究对象，这里假设是英文字母表的前 20 个字母，(a~t)；
- (2) 文本没有空格，没有标点符号；
- (3) 未知片段出现的频率服从 Gauss 分布；
- (4) 由于人类技术限制，替换和插入的字母是随机的，并且每个错误只涉及一个字母，错误是独立发生的；
- (5) 每个字母发生错误的概率是 1/5；
- (6) 每种错误（删失错误、插入错误、替换错误）发生的概率都是 1/3；
- (7) 对于问题一，片段长度为 15；对于问题二，希望找到的片段的长度在 15 - 21 个字母之间（参考第一阶段的问题要求）。

3.2 名词定义

文章中一些自定义术语描述：

片段库 1：针对问题一构造的片段库，长度固定为 15；

片段库 2：针对问题二构造的片段库，长度随机地在 15~21 之间；

正确文本：由片段库生成的文本，不存在错误；

错误文本：接受正确文本后出现记录错误（替换、插入、删失）的文本；

目标片段：最终希望找出的片段；

中心片段：聚类后中心点对应片段；

非中心片段：聚类后非中心点对应片段；
 中心子串：模型二中目标片段的子片段；
 非中心子串：模型二中非目标片段的子片段；

3.3 符号说明

表 3-1 符号说明

符号	说明
w	字母代号
S	字母片段
S_g	目标片段
S_r	相似片段
S_{sub-g}	中心子串
$S_{un-sub-g}$	非中心子串
L	片段长度
L_g	目标片段长度
L_i	一段文本的长度
L_{sub}	子串长度
L_{sub-g}	目标片段子串长度
i	字母序号 $i \in (1 \sim L)$
n	片段数
k	片段中不同字母个数
P_s	片段频率
P_w	字母频率
$\overline{P_w}$	片段中的字母频率均值
P_{w*}	字母*的频率, $* \in (a \sim t)$
σ^2	片段中的字母频率方差
d	距离
d_L	编辑距离
d_H	汉明距离
Sim	相似度
D_i	s_i 处的密度指标
D_{c1}	s_{c1} 处的密度指标
r_a	s_i 的半径
r_b	D_i 减小领域的半径
q	p 的近似值

四、问题一建模求解

4.1 模型一流程

模型一的流程见图 4-1。

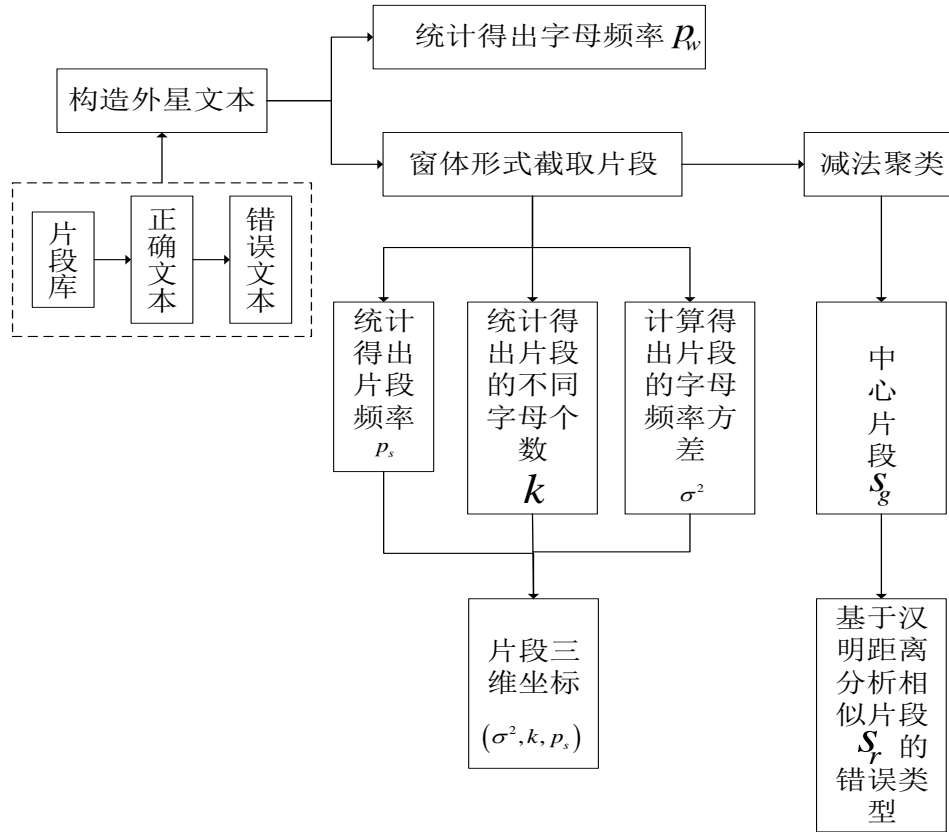


图 4-1 模型一流程图

Step(1) 建立片段库，其实是构造“虚拟词典”，该词典由服从 Gauss 分布的字母序列片段及该片段的概率构成，我们将其称为片段库。为获取满足条件的文本，片段库长固定为 15，由计算机按概率约束随机产生 50 个片段，并且规定每个片段可能在文中出现的概率值，概率值服从 $N(0,1)$ 的 Gauss 分布，形成片段库。

Step(2) 构造正确的外星文本，由这个片段库和 $N(0,0.5)$ 概率分布生成 30 段文本，按照要求，每段的长度是随机的，并且在 5000~8000 之间，由此得到正确文本。

Step(3) 构造错误文本，按“每个字母有五分之一的概率发生错误”的要求，并且每种错误发生的概率都是 1/3（假设）的要求构造出错误的文本，以此来作为本文的研究对象。

Step(4) 统计每个字母在文本中的频率 p_w ，并计算均值 $\overline{p_w}$ 和方差 σ^2 。

Step(5) 截取文本中的全部片段，并基于马尔可夫模型计算每个片段的概率 p_s 。

Step(6) 计算片段的三维坐标 (σ^2, k, p_s) ，其中 k 为片段中不同字母的个数。

Step(7) 减法聚类，找出中心片段，即目标片段 s_g 。

Step(8) 进一步地，根据中心片段，基于汉明距离分析可能出现错误的片段和出现错误的类型。

4.2 模拟外星文本

4.2.1 建立片段库 1

不可否认语言的产生是根据一定的规则的，并且从经验的角度出发，文本中某词或字（本文中相当于一个片段）出现频率一般服从 Gauss 分布。基于此通过 Java 语言随机合成长度为 15 个字母的片段 S_i ($0 < i \leq 50$)，共 50 个，20 个字母在片段库中呈高斯分布。再通过 $N(0,0.5)$ 的高斯的概率密度函数产生一组概率 $P(S_i)$ ，将概率随机分配给这些片段 S_i 。

再将 $P(S_i)$ 的概率值顺序的分给这 50 个片段，意味着这 50 个片段将按所给自身的概率出现在文本当中，由此，我们构造了片段库 1，表示为 D，用于检验结果。

片段的概率计算表示为：

$$P(S_i) = \frac{1}{\sqrt{2\pi}\delta} e^{\left(-\frac{(x-\mu)^2}{2\delta^2}\right)}, \quad i \in [0, 50] \quad (4-1)$$

片段的概率分布图见图 4-2，计算机生成的片段库 1 见附录二。

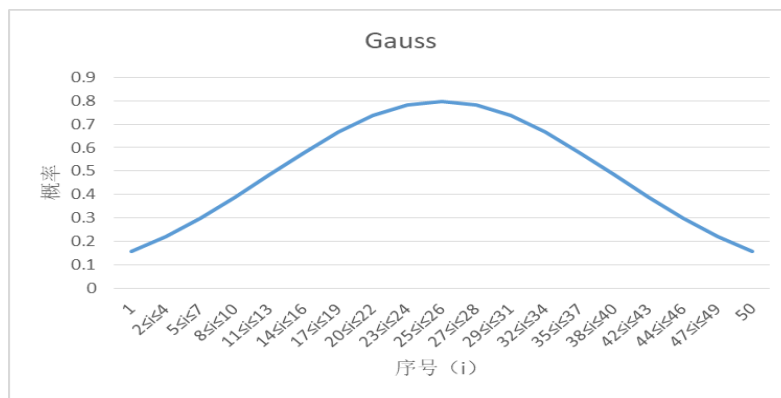


图 4-2 片段的概率分布图

4.2.2 生成正确文本

接下来我们根据 $N(0,0.5)$ 概率分布，利用片段库 D 随机生成 30 段正确文本，特点有：

- (1) 每段的长度是随机的，并且在 5000~8000 之间；
- (2) 没有记录错误。

4.2.3 生成错误文本

在正确文本的基础上，用 Java 重写文本，该重写随机加入了以下三个错误：

- (1) 删失错误：丢失了某个字母；
- (2) 插入错误：新增了原本不存在的字母；
- (3) 替换错误：某个字母被篡改成了其他的字母。

并且每个字母有 20% 的概率发生错误，并且每种错误发生的概率都是 1/3，大致流程见图 4-3。生成的错误片段即文星文本，即研究对象。

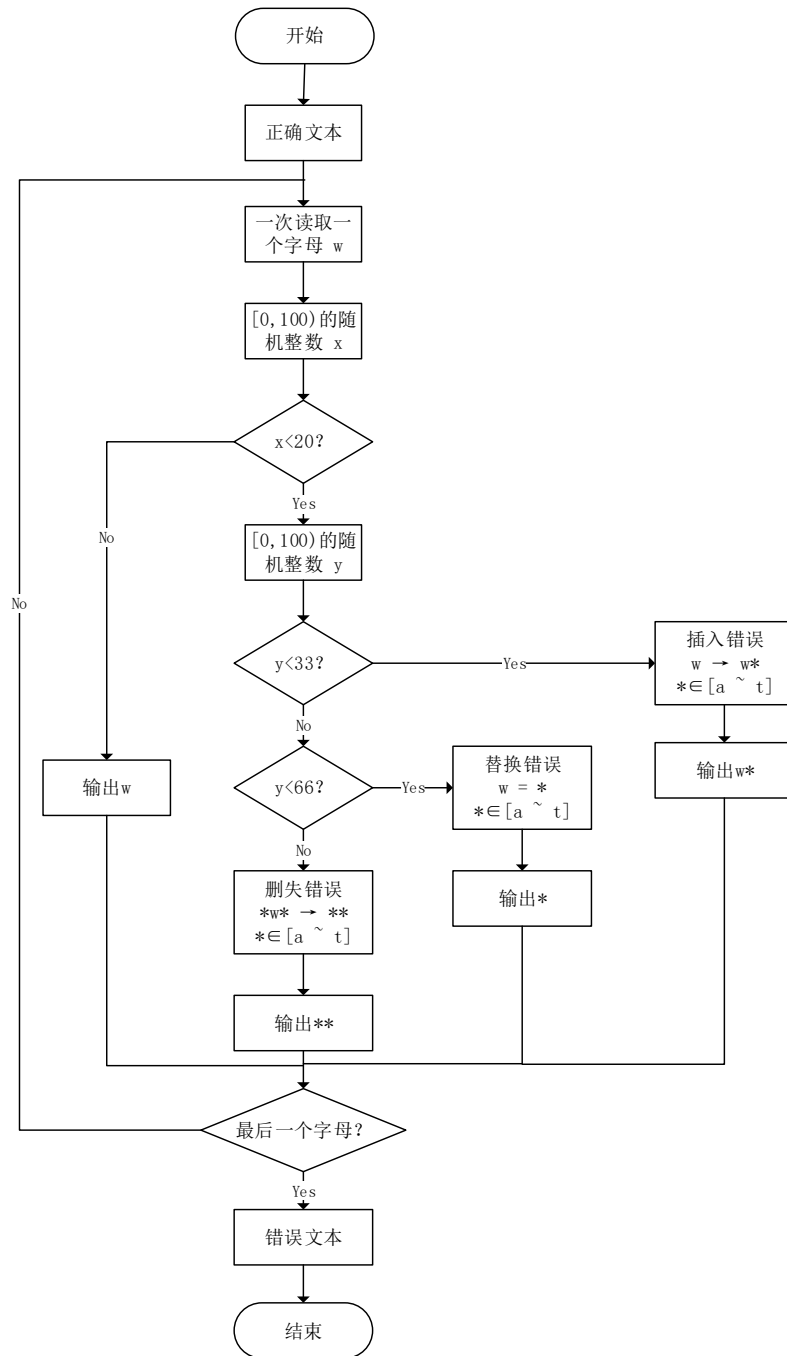


图 4-3 由正确文本生成错误文本流程图

4.3 基于统计获取全部固定长度片段

4.3.1 片段截取

我们需要从已获取文本中截取片段，而且要保证截取的片段包含了所有可能性。在这里我们以“窗体”的形式来截取片段，窗体的长度固定为 15 位字母的长度，从外星文本的首位字母开始，每截取一次，窗体右移一位，以此类推，直至文本的倒数第 18 位字母的位置为止。示意图如图 4-4 所示：第一次取红色括号里的内容，截取完成后右移一位，截取蓝色括号里的内容。

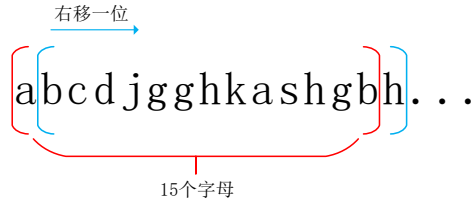


图 4-4 “窗体”形式截取片段示意图

由此获得了所有可能存在的片段，再对片段进行概率统计，见下一小节。

4.3.2 基于马尔可夫链的初步统计

在语言处理中，一种处理文本的思路，就是计算出某个片段在语言中出现的概率。那么，在处理已获取的文本我们将截取长度为 15 的所有可能存在的片段，并利用马尔可夫链计算其出现的概率，根据概率值进行降序排序，再进入下一步的筛选。

那么我们需要对源文本进行初步统计，取出所有可能的定长片段，并利用 n 元语法模型（N-Gram）中的 2 元语法模型，即马尔可夫链（Markov Chain, MC），获得马尔可夫一步转移概率，再根据概率值降序输出所有片段及其对应的概率。

对于一个片段，由 m 个字母构成，令 s 表示片段， w 表示单个字母，就有：

$$s = w_1 w_2 w_3 \cdots w_m$$

其概率可以表示为：

$$\begin{aligned} p(s) &= p(w_1) p(w_2 | w_1) p(w_3 | w_1 w_2) \cdots p(w_m | w_1 w_2 \cdots w_{m-1}) \\ &= \prod_{i=1}^m p(w_i | w_1 \cdots w_{i-1}) \end{aligned} \quad (4-2)$$

式 4-2 中，产生第 i 个基元的概率是由前面的 $i-1$ 个字母 $w_1 w_2 \cdots w_{i-1}$ 决定的。一般地，我们将前 $i-1$ 个字母 $w_1 w_2 \cdots w_{i-1}$ 成为第 i 个字母地“历史（History）”，那么随着历史长度地增加，不同的历史数目按指数级增长，片段的概率也就越小。如果历史长度为 $i-1$ ， L 为片段的容量，那么就会有 L^{i-1} 种不同的历史，而为了计算第 i 个字母的概率，就必须考虑所有 L^{i-1} 种不同的历史情况下，产生第 i 个字母的概率。也就是说，这样的模型中有 L^i 个自由参数 $p(w_i | w_1 \cdots w_{i-1})$ ，这就导致几乎不可能从数据集中正确地估计出这些参数。为了解决这个问题，提出了 n 元语法模型（N-Gram）解决方案。我们假设，每个字母只与前面相邻的 $m-1$ 个字母有关，这样， $p(w_i | w_1 \cdots w_{i-1})$ 可以近似为 $p(w_i | w_{i-m+1} \cdots w_{i-1})$ ，则式 2 可以近似等于：

$$p(s) = \prod_{i=1}^m p(w_i | w_{i-m+1} \cdots w_{i-1}) \quad (4-3)$$

式 4-2 式多个概率连乘得到的，但是在从文本中统计这些概率的时候，并不能保证每个概率都不为 0，如果有一个概率为 0， $p(s)$ 就会等于 0，则会出现数据稀疏现象，为解决这个问题，我们取 $m=2$ ，即二元语法模型（实质是一个马尔科夫链），具体表示每个字母只与前面仅邻的一个字母有关，所有字母出现的概率 $p(w_i | w_{i-1})$ 就组成了马尔可夫一步转移概率。式 4-3 就近似变成：

$$p(s) = \prod_{i=1}^m p(w_i | w_{i-1}) \quad (4-4)$$

基于式 3 进行所有片段的概率计算与统计，所有结果按概率值降序输出每个片段及其概率，得到初步片段统计结果，见附录二。

4.3.3 计算字母频率

在统计语言学中有一个重要的特点是研究每一个基元的特点，知己知彼百战不殆，本文研究统计每个字母的分布，即在外星文本中统计每个字母出现的频率 p_w ，统计结果见附录二。

4.3.4 确定三维坐标

首先寻找每个片段的属性特征，在前面的工作中已经得出了每个片段的两个属性，一方面是每个片段的马尔可夫统计概率，作为第三维坐标；另一方面是每个片段字母频率的方差 σ^2 作为第一维坐标。方差公式如下：

$$\sigma^2 = \frac{\sum_{i=1}^L (p_{wi} - \overline{p_w})^2}{L} \quad (L=15) \quad (4-5)$$

式中 p_{wi} 为字母 wi 出现的频率， $\overline{p_w}$ 为每个片段的字母概率均值， L 为片段长度。已知每个字母的概率分布则可得出每个片段的字母概率均值 $\overline{p_w}$ ：

$$\overline{p_w} = \frac{1}{L} \cdot (p_{w1} + p_{w2} + \dots + p_{wL}) \quad (4-6)$$

例如片段 s 为 “abcdefghijklmno”，则字母概率均值 $\overline{p_w}$ 为：

$$\overline{p_w} = \frac{1}{15} \cdot (p_{wa} + p_{wb} + \dots + p_{wo}) \quad (L=15) \quad (4-7)$$

另以片段中出现的不同符号个数 k 来作为第二维坐标值，例如片段 s 为 “abcdabcdeaaafgtt”，出现的不同字母为 “abcddefgt”，则该片段的 k 值为 8。

那么我们就可以确定每个片段的三维坐标为：(σ^2, k, p_s)。

4.4 减法聚类

减法聚类是一种用于估计一组数据中的聚类个数记忆以及聚类中心位置的快速单次算法 (One Pass)。由减法聚类算法得到的聚类估计可以用于初始化那些基于重复优化过程的模糊聚类以及模型辨识方法。

减法聚类方法将每个数据点作为可能的聚类中心，并根据各个数据点周围的数据点密度来计算改点作为聚类中心的可能性。被选为聚类中心的数据点周围具有最高的数据点密度，同时该数据点附近的数据点被排除作为聚类中心的可能性；在选出第一个聚类中心后，从剩余的可能作为聚类中心的数据点中，继续采用类似的方法选择下一个聚类中心。这一过程一直持续到所有剩余的数据点作为聚类中心的可能性低于某一阈值时。

考虑 M 维空间的 n 个数据点 (片段) (s_1, s_2, \dots, s_n) 。不失一般性，假设数据点已经归一化到一个超立方体。由于每个数据点都是聚类中心的候选者，因此，数据点 x_i 处的密度指标定义为

$$D_i = \sum_{j=1}^n \exp \left(\frac{\|s_i - s_j\|^2}{(r_a/2)^2} \right) \quad (4-8)$$

这里 r_a 为一个正数。显然，如果一个数据点有多个邻近的数据点，则该数据点具有高密度值。半径 r_a 定义了该点的一个邻域，半径以外的数据点对该点的密度指标贡献甚微。

在计算每个数据点密度指标后，选择具有最高密度指标的数据点为第一个聚类中心，令 s_{c1} 为选中的点， D_{c1} 为其密度指标。那么每个数据点 s_i 的密度指标公式

$$D_i = D_i - D_{c1} \exp\left(\frac{\|s_i - s_{c1}\|^2}{(r_b/2)^2}\right) \quad (4-9)$$

修正。其中 r_b 为一个正数。显然，靠近第一个聚类中心 s_{c1} 的数据点的密度指标显著减小，这样这些点不太可能成为下一个聚类中心。常数 r_b 定义了一个密度指标函数显著减小的邻域，常数 r_b 通常大于 r_a ，以避免出现相距很近的聚类中心。一般取 $r_b = 1.5 r_a$ 。

修正了每个数据点的密度指标后，选定下一个聚类中心 s_{c2} ，再次修正数据点所有密度指标。该过程不断重复，直到产生足够多的聚类中心，也可以根据一定的条件自动确定聚类的个数。

该模型中对截取出来的所有片段进行“减法聚类”，主要目的就是找到中心片段即目标片段 s_g 。聚类使用 matlab 进行计算，调用“subclust(X, radii, Xbounds, options)”函数就可以实现，模型中设定的 X 为三维数据，即 (σ^2, k, p_s) 三维坐标点， σ^2 为片段的字母频率方差，其代表每一个片段相对比总的片段而言其构成内容的独特性； k 为片段的不同字母个数，代表每一个片段中所含字母的丰富性； p_s 为片段频率，代表每一个片段在总片段中所占的比重；综上所述，我们选择这三个指标作为每一个片段的特征来进行聚类是比较合理的。

与片段库片段进行比较，验证聚类所得中心片段的准确性，见表 4-1。

表 4-1 聚类所得中心片段与片段库片段对比

序号	片段库片段	目标片段 s_g	σ^2	k	p_s
1	nlllffffifdbkl1l	nlllffffifdbkl1l	0.06	7	0.666667
2	sqsnnffldccnnoo	sqsnnffldccnnoo	0.05	7	0.566667
3	mkkkjsjjjndhgc	mkkkjsjjjndhgc	0.06	8	0.566667
4	kojiccckfjbdkec	kojiccckfjbdkec	0.05	10	0.5
5	kqfffnaoofrpiek	kqfffnaoofrpiek	0.06	6	0.466667
6	hjjifofoqnfkehc	hjjifofoqnfkehc	0.06	7	0.466667
7	csdgkkkpgfgjhig	csdgkkkpgfgjhig	0.05	7	0.433333
8	eejjcjimggmef	eejjcjimggmef	0.06	9	0.433333
9	bplqoteeqqhrccf	bplqoteeqqhrccf	0.06	6	0.4
10	kkkkbjrrmbtttoh	kkkkbjrrmbtttoh	0.06	10	0.5
11	clqbtjfmjssmdd	clqbtjfmjssmdd	0.06	9	0.466667
12	pfffjbrrnqhrsrr	pfffjbrrnqhrsrr	0.06	10	0.533333
13	kqfffnaoofrpiek	kqfffnaoofrpiek	0.06	11	0.433333
14	klorrrrfnnikppi	klorrrrfnnikppi	0.06	9	0.4
15	hlmlllkoksknnn	hlmlllkoksknnn	0.05	9	0.433333

注：红色：表示片段不相同的部分

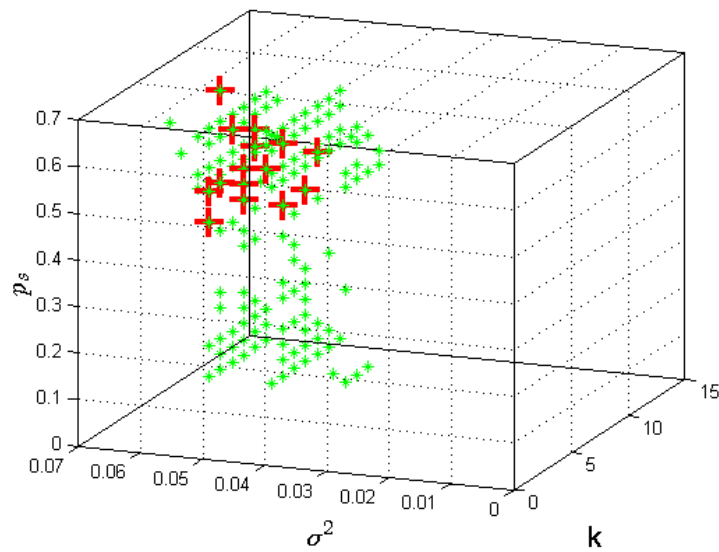


图 4-5 模型一中心片段空间图

4.5 基于中心片段和汉明距离分析错误类型

以上部分已经找出了目标片段，本模型还能继续找出错误片段，以及分析出他们的错误类型。考虑到错误类型有三种（删失错误、插入错误和替换错误），其中替换错误是最容易找出来的，而删失错误和插入错误的发生会导致文本长度发生改变，通过这个特点，我们进行了以下步骤，分析错误片段的错误类型，若时间允许，还能将其进行修正。

4.5.1 中心片段与非中心片段汉明距离

从聚类中获得 15 个中心片段，分别将 15 个片段与其同一簇中的非中心片段作相似对比，计算每个中心片段与其簇中的其他片段的汉明距离，用以判断聚类出的片段是否满足相似性要求，即这些非中心片段是否由中心片段发生错误转换而来。

汉明距离计算了两个等长字母片段 s_1 与 s_2 之间对应位置上不同字母的个数。它强调片段中字母位置的一一对应，且比较的两个片段对象必须等长，否则距离不成立。此模型中取出的片段长度皆为 15，应用此方法可准确计算出 s_1 、 s_2 对应位置上不同字母个数。如图 4-6 所示片段 s_1 与 s_2 在红色箭头标示处的字母不同，其他位置处字母均能一一对应，存在 3 个字母不同则得出距离 $d=3$ 。

S_1	a c m q e l b c g i j n t o s	
	↓ ↓ ↓ ↓ ↓ ↓	$d=3$
S_2	a r m o e l b c g k j n t o s	

图 4-6 S_1 、 S_2 片段位置对比

汉明距离已知则可根据 4-10 计算片段间的相似度。式中 Sim 为相似度， d 为距离， L 为片段长度，此模型中设定 $L=15$ 。

$$Sim = 1 - \frac{d}{L} \quad (4-10)$$

根据公式 4-10，依次求出汉明距为 1~15 时的片段相似度(结果保留两位小数)，见表 4-2：

表 4-2 相似度计算结果

汉明距离 dH	相似度 Sim (结果保留两位小数)
1	0.93
2	0.86
3	0.80
4	0.73
5	0.67
6	0.60
7	0.53
8	0.47
9	0.40
10	0.33
11	0.27
12	0.20
13	0.13
14	0.06
15	0.00

由上述相似度计算结果可知汉明距离越大，相似度越低；汉明距离越小，相似度越高。表 4-3 为从 15 个中心片段中取出片段 s_1 与其部分相似片段 $s_{r1} \sim s_{r2}$ 的相似度比较。由对比可见两片段在不考虑位置关系时，其字符内容差异不明显，而用汉明距离计算其相似度则会因相似度较低舍弃这些内容相似的片段。故此模型中保留内容相似片段，将其视为可能发生记录错误的片段。判定聚类得到的中心片段就是模型最终要找出的目标片段，其簇中的非中心片段皆为内容相似度较高的相似片段。

表 4-3 Sg1 及部分相似片段 Sr 相似度

序号	目标片段 Sg1	相似片段 Sr	汉明距离 dH	相似度 Sim
1		n l l l f f f i f d b k l l p	1	0.93
2		d n l l l f f f i f d b k l l l	9	0.40
3		a n l l f f f i f d b k l l l	2	0.86
4		n l l l f f s q i d b k l l l	3	0.80
5	n l l l f f f i f d b k l l l	n l n i l l f f f i f d b k l l l	6	0.60
6		p l l l f f f i f d b k l l l p	8	0.47
7		m n l l l f f f i f d b k l l p	5	0.67
8		i g n l l f f f i f d b k l l l	4	0.73
9		i o n l l l f f f i f d b k l l l	10	0.33

注：绿色：可能发生缺失，并补入缺失部分（此部分不属于原片段字符）；

红色：可能发生替换、插入或缺失；

黄色：可能发生替换；

紫色：可能发生插入

其他黑体部分为原片段中本含字符，不发生错误。

4.5.2 错误类型分析

由上述汉明距离及相似度计算结果知汉明距离越大，相似度越低；汉明距离越小，相似度越高。若根据此相似度筛选相似片段，判定相似度较高的片段是由目标片段发生错误转换而来，则很容易将由于有插入或缺失错误导致片段位置关系发生移动的相似片

段舍弃。为减小这一问题产生的影响，引入编辑距离验证聚类簇中非中心片段与中心片段的相似程度，依据片段中字母改变的位置判断选出的相似片段，即非中心片段中有可能发生的错误类型：替换、插入、缺失。

假设同一位置仅考虑一种错误，不发生同时缺失、插入，也不发生同时插入并替换。此时根据汉明距大小分两种情况分析片段发生的错误类型。模型中设定 $d < 8$ 为小汉明距， $d \geq 8$ 为较大汉明距离。

(1) 汉明距 $d < 8$ (1~7)

汉明距较小时，假设片段只发生字母替换，此时不改变字母序列位置，仅替换位置产生汉明距。若发生插入或替换且在片段靠右位置，此时发生的错误对片段字母位置影响较小，其汉明距也相对较小。如图 4-7 所示，只发生替换错误时， s_{r1} 与 s_{g1} 对比有 3 个位置的字母改变，其间汉明距为 3。

S_{g1}	a	c	m	q	e	l	b	c	g	i	j	n	t	o	s
	↓	↓	↓	↓				↓					
S_{r1}	a	r	m	o	e	l	b	c	g	k	j	n	t	o	s

d=3

图 4-7 替换错误

(注：蓝色箭头：表示对应位置字母相同；红色箭头：表示对应位置字母不同)

如图 4-8, 插入字母的位置位于片段右端，在第 10 与第 11 位置中间插入字母 **k**, 此时其后字母 (i, j, n, t, o, s) 位置右移, 产生的右移导致汉明距变大为 6。若插入位置越靠右，发生位置移动的字母就越少，对整个字符串位置影响就越小，其间汉明距也随时变小。汉明距越小，相似度越高。

S_{g1}	a	c	m	q	e	l	b	c	g	i	j	n	t	o	s
	↓								↓	↓			
S_{r2}	a	c	m	q	e	l	b	c	g	k	i	j	n	t	o

d=6

图 4-8 插入位置于片段右端的插入错误

同样地，在片段右端发生删失错误时，删失字母位置其后的字母将左移。如图 4-9, 位置 10 处的字母 **i** 删失, 其后的字母 (j, n, t, o, s) 左移一位, C 为紧临片段的首字母, 产生的左移导致汉明距变大为 6。若删失位置越靠右，发生位置移动的字母就越少，对整个字符串位置影响就越小，其间汉明距也随时变小。汉明距越小，相似度越高。

S_{g1}	a	c	m	q	e	l	b	c	g	i	j	n	t	o	s
	↓							↓	↓				
S_{r3}	a	c	m	q	e	l	b	c	g	j	n	t	o	s	c

d=6

图 4-9 删失位置于片段右端的删失错误

汉明距离较小时可判断为两种情况。第一，片段可能只发生替换；第二，片段发生插入或删失错误的位置于片段右端。

(2) 汉明距 $d \geq 8$ (8~15)

汉明较大时，假设片段中出现替换错误的同时出现删失或插入错误，且此时插入或删失的字母位置于片段左端，插入时其后字母均右移，删失时其后字母均左移，不考虑字母缺失的同时有字母插入。如图 4-10, 在首位置插入字母 **g**, 则其后字母

(cmqelbcgijntos) 皆右移, 此时两片段只有一个字母有差异, 相似度极高, 而其汉明距达到 14。由此判断片段可能发生插入错误且插入位置于片段左端。

S_{g1} a c m q e l b c g i j n t o s
 S_{r4} g a c m q e l b c g i j n t o $d=14$

图 4-10 插入位置于片段左端的插入错误

如图 4-11, 首字母 a 删失导致其后字母皆左移, p 为紧临片段的首字母。删失位置于片段右端使大部分字母错位。此时两个片段内容相似度较高, 而其汉明距却达到 14。由此判断片段可能发生删失错误且错误位置于片段左端。

S_{g1} a c m q e l b c g i j n t o s
 S_{r5} c m q e l b c g i j n t o s p $d=14$

图 4-11 删失位置于片段左端的删失错误

汉明距离较大时可判断为两种情况。第一, 片段可能发生插入错误且位置于片段左端; 第二, 片段可能发生删失错误且位置于片段左端。此情况下可能同时发生替换错误, 暂不考虑其中。

4.5.3 聚类结果相似度验证

聚类后的非中心片段皆为中心片段的相似片段, 由于汉明距离只考虑片段的位置, 此部分引入编辑距离与其对比验证聚类结果。编辑距离记录了将字母片段 s_1 转换成另一个片段 s_2 所需的最少编辑操作次数, 最大限度考虑片段内容相似, 不考虑位置关系, 这种方法符合模型聚类初衷。如表 4-4, 取出的是聚类后的一个中心片段 s_{g1} 及其簇中部分相似片段。两片段内容相似度较高时编辑距离较小, 汉明距离可能大可能小。聚类后取出的相似片段与中心片段的编辑距离皆小于阈值 4, 表明聚类后的每个簇中相似片段的相似度都很高, 聚类出的结果符合假设。

表 4-4 S_{g1} 及部分相似片段 S_r 的 dL 、 dH

序号	目标片段 S_{g1}	相似片段 S_r	编辑距离 dL	汉明距离 dH
1		n l l l f f f i f d b k l l p	1	1
2		d n l l l f f f i f d b k l l l	2	9
3		a n l l l f f f i f d b k l l l	2	2
4		n l l l f f s q i d b k l l l	3	3
5	n l l l f f f i f d b k l l l	n l n i l l f f f i f d b k l l l	4	6
6		p l l l f f f i f d b k l l l p	3	8
7		m n l l l f f f i f d b k l l p	3	5
8		i g n l l f f f i f d b k l l l	3	4
9		i o n l l l f f f i f d b k l l l	4	10

注: 绿色: 可能发生缺失, 并补入缺失部分 (此部分不属于原片段字符);

红色: 可能发生替换、插入或缺失;

黄色: 可能发生替换;

紫色: 可能发生插入

其他黑体部分为原片段中本含字符, 不发生错误。

五、问题二建模求解

5.1 模型二流程

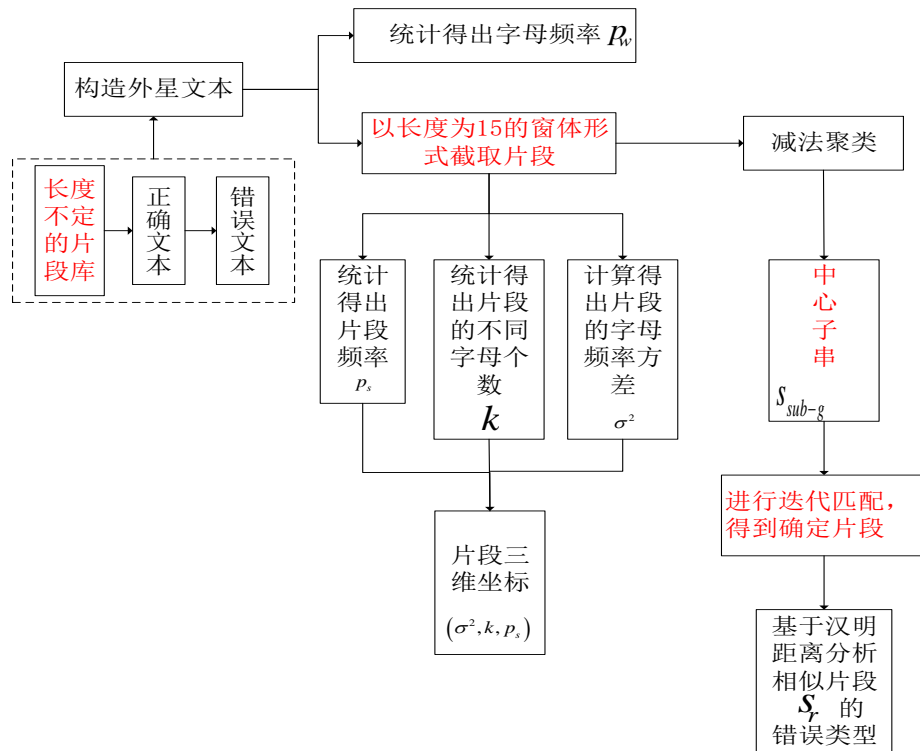


图 5-1 模型二流程图

模型二的流程图如图 5-1 所示，红色部分为在模型一基础上改进的部分。

Step(1) 建立片段库，与模型一的区别是片段库的片段是不定长的，长度在 15~21 之间。

Step(2) 构造正确文本，同模型一。

Step(3) 构造错误文本，同模型一。

Step(4) 统计每个字母在文本中的频率 p_w ，并计算频率均值 $\overline{p_w}$ 和方差 σ^2 ，同模型一。

Step(5) 以最小子串长度（15）为窗体宽度，截取文本中的全部片段，并基于马尔可夫模型计算每个片段的概率 p_s 。

Step(6) 计算片段的三维坐标 (σ^2, k, p_s) ，同模型一。

Step(7) 减法聚类，找出中心子串 s_{sub-g} ，即目标片段 s_g 的子串。

Step(8) 将中心子串 s_{sub-g} 代入外星文进行迭代匹配，获取真正的目标片段。

Step(9) 进一步地，根据目标片段，基于汉明距离分析可能出现错误的片段和出现错误的类型。

5.2 模拟外星文本

在问题一的求解过程中，我们构造了长度均为 15 的片段作为片段库 1 的元素，字母分布服从高斯分布，在问题中我们仍用同一个方法构造符合要求的片段库 2，唯一的区别是片段库中每个片段的长度是随机的、不固定的，范围在 15~21 之间。

建立片段库 2 后，接着产生正确片段，再用正确的片段按错误概率要求生成错误的片段，其方法同模型一。

5.3 基于统计获取全部片段

5.3.1 片段截取

片段长度不固定，要如何以窗体形式截取片段是问题二的难点，这也是我们的模型二首要解决的问题。常规的方法是寻找所有可能的片段出来，如规定窗体长度为 15 进行截取，长度为 16 再截取一次，长度为 17……长度为 21 再截取，然后根据一些算法进行筛选，得出最后的目标片段，如此一来，数据量是模型一的 6 倍，数据空间增大，对于后面的算法，对计算机的硬件条件要求较高，运行速度慢，空间复杂度较低，从时间复杂度的角度出发无疑是不可行的。

为解决这个问题，思考得出，按上面的方法截取所有片段无疑会产生太多的冗余片段，导致数据集按倍数增大，原因是存在很多短片段是很多长片段的子集，即很多长片段其实已经包含了很多短片段。例如有下面一段外星文本：

hacbdfttesmmnllaabcbdefgdabcbdefgdafntesmmnllaabcb

从位置 1 开始截取长度为 15 的片段 s_1 = “hacbdfttesmmnll”，从同一位置截取长度为 19 的片段 s_2 = “hacbdfttesmmnllaabc”，那么更长的片段 s_2 就已经包含了更短的片段 s_1 ， s_1 是 s_2 的子串。

从这个思路出发，假设 s_2 是我们要寻找的目标片段，那么我们截取目标片段 s_2 的最小长度的子串，根据 3.1 节的模型假设，最小子串的长度为 15，在该例子中也就是 s_1 。那么在本步骤中，我们只截取长度为 15 的片段，截取下来的片段称之为子串。

截取所有子串之后，经过聚类得到的是目标片段的子串，设目标片段的长度为 L_g ，目标片段的子串的长度为 L_{sub-g} ，那么 $L_{sub-g} \leq L_g$ 。

该步骤获取的只是子串，还需要 5.5 节的改进，进一步确定目标片段（代入外星文本匹配搜索实际的片段）。

5.3.2 基于统计确定三维坐标

本步骤同模型一对外星文本进行统计，获得每个字母在文本中的频率 p_w （并计算字母频率的均值 $\overline{p_w}$ 和方差 σ^2 ）、每个片段的概率 p_s 、片段中不同字母的个数 k 。由此获得片段的三维坐标 (σ^2, k, p_s) 。

5.4 减法聚类

减法聚类将每个数据点作为可能的聚类中心，并根据各个数据点周围的数据点密度来计算该点作为聚类中心的可能性。被选为聚类中心的数据点周围具有最高的数据点密度，同时该数据点附近的数据点被排除作为聚类中心可能性；在选出第一个聚类中心后，在剩余的可能作为聚类中心的数据点中，继续采用类似的方法选择下一个聚类中心。这一个过程一直持续到所有剩余的数据点作为聚类中心的可能性低于某一阈值时为止。

该模型中对截取出来的所有片段进行“减法聚类”，主要目的就是为了找到中心子串 s_{sub-g} ，即目标片段的子片段，找到中心子片段后再进行搜索即可找出目标片段。聚类使用 matlab 进行计算，调用“subclust(X,radii,Xbounds,options)”函数就可以实现，模型中设

定的 X 为三维数据，即 (σ^2, k, p_s) 三维坐标点， σ^2 为片段的字母频率方差，其代表每一个片段相对比总的片段而言其构成内容的独特性； k 为片段的的不同字母个数，代表每一个片段中所含字母的丰富性； p_s 为片段频率，代表每一个片段在总片段中所占的比重；综上所述，我们选择这三个指标作为每一个片段的特征来进行聚类是比较合理的。

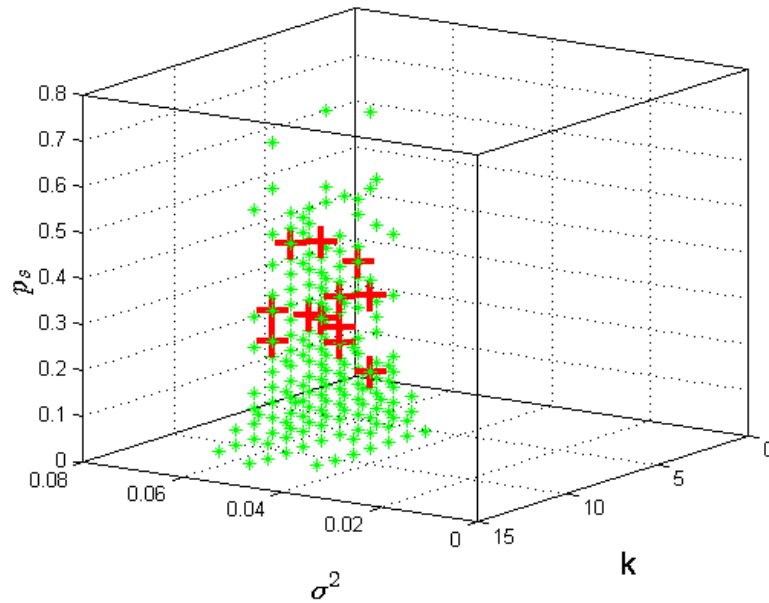


图 5-2 模型二中心子串空间图

5.5 迭代匹配搜索目标片段

5.3.1 小节模型二的片段截取只截取了片段的子串，5.4 节的聚类中心其实是目标片段的子段，不是我们的最终目标

搜索目标片段的过程其实就是以中心子串 s_{sub-g} 为基准片段在外星文本中进行匹配的一个过程，其过程示意图如图 5-3 所示。



图 5-3 迭代匹配搜索示意图

首先，第一个中心子串作为初始基准片段在外星文本中进行匹配，匹配到一模一样的片段时，便将该被匹配片段的字母长度加一（L=16）并记录下来，直至整个外星文本进行一遍匹配为止。

然后，统计出记录片段在外星文本中的出现频率，将所有记录片段的出现频率与初始基准片段的频率（即中心子串频率 $p(s_{sub-g})$ ）的一半进行比较，出现频率大于 $\frac{1}{2}p(s_{sub-g})$ 的记录片段将作为新的基准片段（新的基准片段频率即为该记录片段的频率）再次被放到外星文本中做匹配，匹配到一模一样的片段时，便将该被匹配片段的字母长度加一（L=17）并记录下来，直至整个外星文本进行一遍匹配为止。

不断重复匹配过程，直至所有记录片段的出现频率都小于基准片段频率的一半时停止匹配，此时我们搜索出的目标片段即为此次匹配中的基准片段。

最后，所有的中心子串重复进行与前面两步相同的匹配过程便找出了所有的目标片段。

与片段库片段进行比较，验证迭代匹配搜索所得目标片段的准确性。

表 4-5 迭代匹配搜索所得目标片段与片段库片段对比

序号	片段库片段及长度	目标片段 s_g 及长度	σ^2	k	p_s
1	qhjjgnergttlgeei(16)	qhjjgnergttlgeei(16)	0.06	8	0.2667
2	ddmmmjjrrmbimnjj(17)	ddmmmjjrrmbimnjj(17)	0.06	10	0.3
3	bbfffcseerrmmmfhndb(19)	bbfffcseerrmmmfhndb(19)	0.05	10	0.3
4	egmksopggppigggmmdppp(20)	egmksopggppigggmmdppp(20)	0.05	8	0.4
5	lldhrjqenrrjsbrrlkked(21)	lldhrjqenrrjsbrrlkked(21)	0.04	10	0.2
6	ffgmmrrssihhefmrtjfff(20)	ffgmmrrssihhefmrtjfff(20)	0.06	10	0.2333
7	bbffffcseerrmmmfhndb(19)	bbffffcseerrmmmfhndb(19)	0.05	9	0.3333
8	ddmmmjjrrmbimnjj(17)	ddmmmjjrrmbimnjj(17)	0.05	9	0.2667
9	hqdjhp pphhbabqkk(16)	hqdjhp pphhbabqkk(16)	0.04	10	0.3667

10	m c j j j j q q q q c h h h o (1 5)	m c j j j j q q q q c h h h o (1 5)	0.06	9	0.4333
11	k k h j f f h k k l l g j d d (1 5)	k k h j f f h k k l l g j d d (1 5)	0.05	10	0.4667
12	p n n n l e e s r r c c i i g h n t t k k (2 1)	p n n n l e e s r r c c i i g h n t t k k (2 1)	0.05	9	0.2333

注：红色：表示片段不相同的部分

5.6 基于目标片段和汉明距离分析错误类型

5.6.1 目标片段与非中心子串汉明距离

获得 12 个长度不一的目标片段后，分别将这 12 个片段与长度皆为 15 的非中心子串作相似对比，计算每个目标片段与非中心子串的汉明距离，用以判断获取的子串是否满足相似性要求。因为汉明距离只针对等长字符串，若子串与目标片段的长度不一，则将子串用符号“-”在其末尾补齐，补齐后的片段长度与目标片段一致。在对比过程中，假设补位上的字符不发生错误，即补上的字母与目标片段对应位置的字母相同，且只取目标片段的前 15 位与子串进行对比。

同模型一，汉明距离已知则可计算片段间的相似度。此模型中仅取目标片段的前 15 位与子串进行对比，对比片段的长度皆为 15，汉明距对应的相似度可见表 4-2。

表 4-6 为从 12 个目标片段中取出一个片段 s_{g1} 与其相似的部分非中心子串 $S_{un-sub-g1} \sim S_{un-sub-g6}$ 的相似度比较，此中心片段长度为 17。由对比可见两片段在不考虑位置关系时，其字符内容差异不明显，而用汉明距离计算其相似度则会因相似度较低舍弃这些内容相似的子串。故此模型中保留内容相似非中心子串，将其视为可能发生记录错误的片段。

表 4-6 S_{g1} 及部分非中心子片段 $S_{un-sub-g}$ 相似度

序号	目标片段 S_{g1}	非中心子串 $S_{un-sub-g}$	汉明距 d_H	相似度 Sim
1	ddmmmjjrrmbimnjj	ddmmfjjrrmbimn--	1	0.93
2		ddmmmjjrrgrmbimn--	6	0.6
3		addmmmjjrrmbimn--	9	0.4
4		ddmmmjjrrmjibim--	4	0.73
5		ddmmnjjrrmbimnb--	8	0.47
6		ddmmmjjrrmbimnc--	3	0.8

注：“-”表示补位；（此表中可定位 jj）；
 绿色：可能发生缺失，并补入缺失部分（此部分不属于原片段字符）；
 黄色：可能发生替换；
 紫色：可能发生插入；其他黑体部分为原片段中本含字符，不发生错误。

5.6.2 错误类型分析

由上述汉明距离及相似度计算结果知汉明距离越大，相似度越低；汉明距离越小，相似度越高。若根据此相似度筛选相似非中心子串，判定相似度较高的子串是由目标片段的子串发生错误转换而来，则很容易将由于有插入或缺失错误导致片段位置关系发生移动的相似子串舍弃。为减小这一问题产生的影响，参照汉明距离的同时观察两片段内容，依据子串中字母改变的位置判断选出的相似子串有可能发生的错误类型：替换、插入、缺失。

假设同一位置仅考虑一种错误，不发生同时缺失、插入，也不发生同时插入并替换。默认补齐位置字母与目标片段相应位置字母相同。此时根据汉明距大小分两种情况分析片段发生的错误类型。模型中设定 $d = (1 \sim L/2)$ (L 为目标片段的长度) 为小汉明距， $d = (L/2+1 \sim L)$ 为较大汉明距离。下文中以长度为 17 的目标片段为例。

(1) 汉明距 $d < 9$ ($1 \sim 8$)

汉明距较小时，假设片段只发生字母替换，此时不改变字母序列位置，仅替换位置产生汉明距。若发生插入或替换且在片段靠右位置，此时发生的错误对子串片段字母位置影响较小，其汉明距也相对较小。如图 5-4 所示，只发生替换错误时， s_{sub-g1} 与 s_{g1} 对比有 2 个位置的字母改变，其间汉明距为 2。

S_{g1} a c m q e l b c g i j n t o s g b
 ↓ ↓ ↓ ↓ d=2
 $S_{un-sub-g1}$ a r m o e l b c g i j n t o s - -

图 5-4 替换错误

（注：蓝色箭头：表示对应位置字母相同；红色箭头：表示对应位置字母不同；“-”代表补位）

如图 5-5，插入字母的位置位于片段右端，在位置 12 与 13 之间插入字母 **k**，此时其后字母（t, o, s, -, -）位置右移，产生的右移导致汉明距变大为 4。若插入位置越靠右，发生位置移动的字母就越少，对整个字符串位置影响就越小，其间汉明距也随时变小。汉明距越小，相似度越高。

S_{g1} a c m q e l b c g i j n t o s g b
 ↓ ↓ ↓ ↓ d=4
 $S_{un-sub-g2}$ a c m q e l b c g i j n **k** t o s -

图 5-5 插入位置于片段右端的插入错误

同样地，在片段右端发生删失错误时，删失字母位置其后的字母将左移。如图 5-6，位置 12 处的字母 **n** 删失，其后的字母（t, o, s, -, -）左移一位，c 为紧临片段的首字母，产生的左移导致汉明距变大为 6。若删失位置越靠右，发生位置移动的字母就越少，对整个字符串位置影响就越小，其间汉明距也随时变小。汉明距越小，相似度越高。

S_{g1} a c m q e l b c g i j n t o s g b
 ↓ ↓ ↓ ↓ d=6
 $S_{un-sub-g3}$ a c m q e l b c g i j t o s - - **c**

图 5-6 删失位置于片段右端的删失错误

汉明距离较小时可判断为两种情况。第一，片段可能只发生替换；第二，片段发生插入或删失错误的位置于片段右端。

（2）汉明距 $d \geq 9$ (9 ~ 17)

汉明距较大时，假设片段中出现替换错误的同时出现删失或插入错误，且此时插入或删失的字母位置于片段左端，插入时其后字母均右移，删失时其后字母均左移，不考虑字母缺失的同时有字母插入。如图 5-7，在首位置插入字母 **g**，则其后字母

（cmqelbcgijntos--）皆右移，此时两片段只有一个字母有差异，相似度极高，而其汉明距达到 17。由此判断片段可能发生插入错误且插入位置于片段左端。

S_{g1} a c m q e l b c g i j n t o s g b
 ↓ ↓ ↓ d=17
 $S_{un-sub-g4}$ **g** a c m q e l b c g i j n t o s -

图 5-7 插入位置于片段左端的插入错误

如图 5-8，首字母 **a** 删失导致其后字母皆左移，**p** 为紧临片段的首字母。删失位置于片段右端使大部分字母错位。此时两个片段内容相似度较高，而其汉明距却达到 17。由此判断片段可能发生删失错误且错误位置于片段左端。

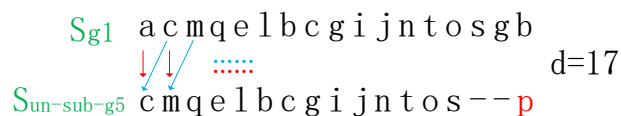


图 5-8 删失位置于片段左端的删失错误

汉明距离较大时可判断为两种情况。第一，片段可能发生插入错误且位置于片段左端；第二，片段可能发生删失错误且位置于片段左端。此情况下可能同时发生替换错误，暂不考虑其中。

六、模型的评价与推广

6.1 模型性能分析

考虑到模型二是由模型一改进而来的，本节对最终的模型（模型二）进行了性能分析，下面从统计、片段截取的数据量、迭代匹配搜索三个方面分析模型的性能。

6.1.1 从统计的角度分析

模型二的前期工作中必须对外星文本做各方面的大量的统计，例如字母出现的频率，马尔可夫链的概率模型等，本文用语法模型中的交叉熵来度量模型的性能，那么需要概率的片段与其估计模型 q 的交叉熵为：

$$H(L, q) = -\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{s_{1n}} p(s_{1n}) \log q(s_{1n}) \quad (6-1)$$

上式中 $q(s)$ 为近似估计 $p(s)$ 的概率分布。

在计算片段概率时，N-Gram 无法直接计算出 $p(s)$ ，模型中运用的马尔可夫模型可以近似估计出 $p(s)$ 。假设外星语言是稳态遍历的随机过程，即 n 趋于无穷大时， $p(s_{1n})$ 为常量，只要 n 足够大，便可近似计算交叉熵：

$$H(L, q) \approx -\frac{1}{n} \log q(s_{1n}) \quad (6-2)$$

交叉熵越小，模型就越能接近真实的概率分布，也就意味着它在应用中的性能越好。

6.1.2 从时间复杂性和空间复杂性角度分析

算法的复杂性是评价算法优劣的重要依据，算法的复杂性有时间复杂性和空间复杂性。对任意给定的问题，设计出复杂性尽可能低的算法是设计算法中追求的一个重要目标；当给定的问题有多种算法时，选择其中复杂性最低者，是选用算法所遵循的一个重要准则。因此，算法复杂性分析对算法的设计或选用有着重要的指导意义和实用价值。

从数据量上看，模型二中子串截取和搜索方法相对于传统方法，同时降低了时间复杂性和空间复杂性，设文本的长度为 L_t ，片段长度为 L ，那么传统截取方法长生的数据量 N 为：

$$N = \sum_{L=L_{\min}}^{L_{\max}} 30(L_t - L) \quad (6-3)$$

模型二基于子串概念截取方法减小了($L_{\max} - L_{\min}$) 倍数据量,同时减小了冗余片段,又通过迭代匹配算法高效而准确的进行全局搜索,找到目标片段,从时间上也大大减少了花费。

6.2 模型的优点

- 1) 模型二适用于具有一定语言规律的真实文本,但本文构造的数据源来自计算机模拟出的外星文本,与实际的外星文本相比,语言的规律性不够充足,从统计的角度来看,规律越多统计得出的结果越接近真实值,所以此模型如果用在真实的语言分析中效果会更好。
- 2) 聚类算法的复杂度只与数据的维度呈线性关系,本文通过对片段的仔细研究,选择出三个特征量既能体现出各片段相较于所有片段的特征,又十分适用于聚类算法,大大减少了聚类算法的开销,提高了模型的计算效率。
- 3) 本文在模型二中模型一的基础上加入了“迭代匹配搜索算法”,以层层递进的方式进行片段匹配,这样在不需要额外匹配的情况下就能搜索出目标字母片段。
- 4) 在查阅自然语言处理、文本信息处理相关资料的基础上,给出了判断一个片段是否具有某种固定含义的两个指标:概率和相似度,从而建立了基于这两个指标的片段检测模型,对客观事实的反应更为恰当。并且模型的可行性较强,适合应用在对语言的研究和学习中
- 5) 本文通过两种相似度算法的比较,找出了其中各为符合该模型的汉明距算法,相对于只尝试一种算法的模型更为精确。

6.3 模型的缺点

- 1) 由于问题的限制,模型中构造出的外星文本语言规律性不强,而真实的语言系统会存在一定的语言规律性,马尔可夫模型适用于真实文本,在构造出来的文本中运用会存在局限性。
- 2) 模型能够高效且较准确的找出错误片段,但由于文本受随机性影响,找出的错误片段数据较多,片段错误类型仅受限于字母替换、删失、插入三种且假设同一片段中同时发生替换、删失的字母个数较少。
- 3) 获取长度不一的目标片段时,改进模型算法使效率提高的同时可能不可避免地会因数据量缩减遗失少部分相似片段。

6.4 模型推广

本模型提出的迭代匹配搜索算法不仅适用于任意文本的字段匹配,在众多数据匹配处理领域都可以发挥很好的作用。此算法在一定程度上降低了研究模型在空间和时间上的复杂度。模型中运用相似度对比算法对文本进行纠错,在此基础上可对不同比对模型的准确程度进行性能分析,作出合理评价。

本模型的使用灵活性大,范围广,模型中提出的各类算法适用于处理大数据文本,不仅可以用于对未知语言文本进行探索,还可应用于当代语言研究中的自然语言处理(Natural Language Processing, NLP)、机器学习(Machine Learning)等领域。模型的提出符合与时俱进的语言研究方式,同时保留了语言系统有一定规律可循的特点,提高了文本分析的正确性与合理性。

参考文献

- [1] M.P.Bhuyan,S.K. Sarma. An N-gram based model for predicting of word-formation in Assamese language[J]. 2019(2), 427-440.
- [2] Masoud Mahdianpari,Mahdi Motagh,Vahdi Akbari,Fariba Mohammadimanesh,Bahram Salehi. A Gaussian random field model for de-speckling of multi-polarized synthetic aperture radar data[J]. 2019(3), 100-230.
- [3] Qing Zhu,Jun Pei,Xinbao Liu,Zhiping Zhou. Analyzing commercial aircraft fuel consumption during descent: A case study using an improved K-means clustering algorithm[J]. 2019(2), 69-882.
- [4] Jihua Zhu,Zutao Jiang,Georgios D. Evangelidis,Changqing Zhang,Shanmin Pang, Zhongyu Li. Efficient registration of multi-view point sets by K-means clustering[J]. 2019(2), 205-218.
- [5] Li G, Deng D, Wang J, et al. Pass-join: A partition-based method for similarity joins[J]. Proceedings of the VLDB Endowment, 2011, 5(3): 253-264
- [6] Wang J, Feng J, Li G. Trie-join: Efficient trie-based string similarity joins with edit-distance constraints[J]. Proceedings of the VLDB Endowment, 2010, 3(1-2):1219-1230
- [7] 江铭虎. 自然语言处理[M]. 高等教育出版社, 2006(12), 231-239.
- [8] 宋继华, 杨尔弘, 王强军. 中文信息处理教程[M]. 高等教育出版社, 2011(6), 13-24.
- [9] 费浦生, 羿旭明. 数学建模及其基础知识详解[M]. 武汉大学出版社, 2007(1), 50-73.
- [10] 戴琳. 概率论与数理统计[M]. 高等教育出版社, 2009(9), 66-72.
- [11] 姜丹. 信息论与编码[M]. 中国科学技术大学出版社, 2006(9), 89-95.
- [12] CSDN 论坛 K-means 聚类算法及其 MATLAB 实现. https://blog.csdn.net/code_caq/article/details/68486668
- [13] 郑凯, 欧阳林艳, 林强等. LCS 算法与编辑距离算法的研究[J]. 信息通信, 2015(5): 22-23
- [14] 周品. 概率与数理统计[M]. 清华大学出版社, 2012 (11), 336-361.
- [15] 邓奋发. MATLAB R2015b 概率与数理统计[M]. 清华大学出版社, 2017 (1), 352-356.

附录一 程序

附录 1.1 问题一程序

```

1、片段库、正确文本(Java)
import java.io.*;
import java.util.*;
public class dictionary {
    public static void getStringRandom(String s,int leng) {
        File file=new File(s);
        String line=null;
        String chars = "abcdefghijklmnopqrstuvwxyz";
        Double[]
={0.3,0.6,0.6,0.7,0.75,0.8,0.85,0.9,0.9,0.95,0.95,0.95,0.9,0.85,0.8,0.75,0.7,0.6,0.4,0.3};
        String val = "";//片段字符串
        char c='\0';
        List<String> pd = new ArrayList<String>();//片段
        List<Double> pr = new ArrayList<Double>();//概率
        try{
            BufferedReader buf = new BufferedReader(new FileReader(file));
            while((line = buf.readLine())!=null){
                for(int i = 0; i < leng; i++) {
                    int m = (int)(Math.random() * 20);
                    if (mingZhong(p[m])) {
                        c=chars.charAt(m);
                    }
                    val +=c;
                }
                pd.add(val);//片段
                pr.add(Double.parseDouble(line)); //概率
                val="";
                if(pd.size()>=50){
                    break;
                }
                if(pd.size()>=50){
                    break;
                }
            }
            buf.close();
            for (int m=0;m<pr.size() ;m++ ) {
                System.out.println(pd.get(m)+":"+pr.get(m));
            }
            StringBuilder sb = new    StringBuilder();
            File file_result = new File("AlienLang.txt");
            PrintWriter output = new PrintWriter(file_result);//创建写对象
            Random rand = new Random();

```

```

        for (int k=0;k<30 ;k++ ) {
            int zishu = rand.nextInt(3001)+5000;//生成 5000—8000 之间的随
机数，包括 8000
            while (true) {
                int ra = (int)(Math.random()*50);
                if (mingZhong(pr.get(ra))) {
                    sb.append(pd.get(ra));
                }
                if (sb.length()>=(zishu-15) && sb.length()<=(zishu+15)) {
                    output.println(sb.toString());
                    sb.delete( 0, sb.length() );
                    break;
                }
            }
        }
        output.close();
        System.out.println("结束");
    }catch(Exception e){
        e.printStackTrace();
    }
}

public static boolean mingZhong(double hr) {
    Random r = new Random();
    int a = r.nextInt(100);//随机产生[0,100)的整数，每个数字出现的概率为
1%
    if(a<(int) (hr*100)){ //前 hr*100 个数字的区间，代表的几率
        return true;
    }else{
        return false;
    }
}

public static void main(String[] args) {
    getStringRandom("pro.txt",15);
}
}

```

2、错误文本(Java)

```

import java.io.*;
import java.util.*;
class ErrorText{
    public static void main(String[] args) throws Exception{
        File file_source = new File("AlienLang.txt");
        File file_result = new File("AlienLang_err.txt");
        InputStreamReader is = new InputStreamReader(new
FileInputStream(file_source));//将输入的字节流转换成字符流
        BufferedReader br=new BufferedReader(is);//将字符流添加到缓冲流
        PrintWriter output = new PrintWriter(file_result);//创建写对象
        String str=null;
        String chars = "abcdefghijklmnopqrst";
    }
}

```



```

try{
    while ((str = br.readLine()) != null){
        StringBuilder sb = new    StringBuilder(str);
        for (int i=0;i<sb.length() ;i++ ) {
            if (mingZhong(0.2)) {
                if (err_type()==1) { //插入
                    int ra2 = (int)(Math.random()*20);
                    String ch=String.valueOf(chars.charAt(ra2));

                    sb.replace(i,i+1,String.valueOf(str.charAt(i))+ch);//sb.append(str.charAt(i)+ch);
                    i++; //下一个字母
                }else if (err_type()==2) { //替换
                    String ch=null;
                    while (true) { //不能替换成同一个字母。例如 b 不能
                        替换成 b

                        int ra2 = (int)(Math.random()*20);
                        ch=String.valueOf(chars.charAt(ra2));
                        if (!ch.equals(String.valueOf(str.charAt(i)))) {
                            break;
                        }
                    }
                    sb.replace(i,i+1,ch);//sb.append(ch);
                }else { //缺失
                    sb.delete(i,i+1); //不包含 i+1
                    i--;
                }
            }
        }
        output.println(sb.toString());
    }
} catch (FileNotFoundException e){
    e.printStackTrace();
}
br.close();
output.close();
System.out.println("结束");
}

public static boolean mingZhong(double hr) {
    Random r = new Random();
    int a = r.nextInt(100); //随机产生[0,100)的整数，每个数字出现的概率为
    1%

    if(a<(int) (hr*100)){ //前 hr*100 个数字的区间，代表的几率
        return true;
    }else{
        return false;
    }
}

public static int err_type() {
    Random r = new Random();

```

```

int flag;
int a = r.nextInt(100);//随机产生[0,100)的整数，每个数字出现的概率为
1%
if(a<33){
    flag = 1;
}else if(a<66){
    flag = 2;
}else {
    flag = 3;
}
return flag;
}
}

```

3、窗体截取(Java)

```

import java.io.*;
import java.util.*;
class interception{
    public static void main(String[] args) throws Exception{
        File file_source = new File("AlienLang_err.txt");
        File file_result = new File("result.txt");
        InputStreamReader is = new InputStreamReader(new
FileInputStream(file_source));//将输入的字节流转换成字符流
        BufferedReader br=new BufferedReader(is);//将字符流添加到缓冲流
        Map<String,Integer> map = new HashMap<String,Integer>();
        String str=null;
        try{
            while ((str = br.readLine()) != null){
                for (int i=0;i<(str.length()-15);i++) {
                    StringBuilder sb = new StringBuilder();
                    for (int j=0;j<15;j++) {
                        sb.append(str.charAt(i+j));
                    }
                    if(map.containsKey(sb.toString())){//看数组中否已有该元素
                        Integer tempInt = (Integer)map.get(sb.toString());//获
取该汉字的次数，并+1
                        tempInt += 1;
                        map.put(sb.toString(), tempInt);//将汉字及其出现次
数重新加入到 map 中，并且会覆盖相同内容的键
                    }else map.put(sb.toString(), 1);//没有该元素，则加入
                }
            }
        }catch(FileNotFoundException e){
            e.printStackTrace();
        }
        PrintWriter output = new PrintWriter(file_result);//创建写对象
        List<Map.Entry<String,Integer>> list = new
ArrayList<Map.Entry<String,Integer>>(map.entrySet());
        Collections.sort(list, new Comparator<Map.Entry<String,Integer>>() {

```

根据 value 排序

```

//降序排序
public int compare(Map.Entry<String,Integer> o1,
Map.Entry<String,Integer> o2) {
    double result = o2.getValue() - o1.getValue();
    if (result > 0)
        return 1;
    else if (result == 0)
        return 0;
    else
        return -1;
}
});
Iterator<Map.Entry<String,Integer>> iter = list.iterator();//获取 List 集合
的迭代器,Map.Entry<K, V>为迭代元素的类型
while(iter.hasNext()){
    Map.Entry<String,Integer> item = iter.next();
    String key = item.getKey();
    Integer value = item.getValue();
    // output.println( key + ":" + (double)value/30);
    output.println((double)value/30);
}
br.close();
output.close();
System.out.println("结束");
}
}

```

4、字母频率(Java)

```

import java.io.*;
import java.util.*;
class Alph_Frequency{
    public static void main(String[] args) throws Exception{
        File file_source = new File("AlienLang_err.txt");
        File file_result = new File("Alph_Fre.txt");
        InputStreamReader is = new InputStreamReader(new
FileInputStream(file_source));//将输入的字节流转换成字符流
        BufferedReader br=new BufferedReader(is);//将字符流添加到缓冲流
        Map<Character,Integer> map = new HashMap<Character,Integer>();
        String str=null;
        int count = 0;//总字符数
        try{
            while ((str = br.readLine()) != null){
                for (int i=0;i<str.length() ;i++ ) {
                    count++;
                    if(map.containsKey(str.charAt(i)) ){//看数组中否已有该元素
                        Integer tempInt = (Integer)map.get(str.charAt(i));//获
取该汉字的次数，并+1

```

```

        tempInt += 1;
        map.put(str.charAt(i), tempInt); //将汉字及其出现次数重新加入到 map 中，并且会覆盖相同内容的键
    }else map.put(str.charAt(i), 1); //没有该元素，则加入
    }
}
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
PrintWriter output = new PrintWriter(file_result); //创建写对象
List<Map.Entry<Character,Integer>> list = new
ArrayList<Map.Entry<Character,Integer>>(map.entrySet());
Collections.sort(list, new Comparator<Map.Entry<Character,Integer>>()
{ // 根据 value 排序
    //降序排序
    public int compare(Map.Entry<Character,Integer> o1,
Map.Entry<Character,Integer> o2) {
        double result = o2.getValue() - o1.getValue();
        if (result > 0)
            return 1;
        else if (result == 0)
            return 0;
        else
            return -1;
    }
});
Iterator<Map.Entry<Character,Integer>> iter = list.iterator(); //获取 List 集合的迭代器, Map.Entry<K, V>为迭代元素的类型
while(iter.hasNext()){
    Map.Entry<Character,Integer> item = iter.next();
    Character key = item.getKey();
    Integer value = item.getValue();
    output.println( key + ":" + (double)value/count);
}
br.close();
output.close();
System.out.println("结束");
}
}

```

5、坐标（注：第三维坐标是第二个代码输出的结果）(Java)

/**

求字符串坐标

*/

```

import java.io.*;
import java.util.*;
import java.text.DecimalFormat;
public class Coor {
    public static void main(String[] args) {

```

```

File read_file=new File(args[0]);
File writeName = new File("dcoor.txt");
File writeAll = new File("adcoor.txt");
String s=null;
try{
    Reader    reader    =    new    InputStreamReader(new
FileInputStream(read_file)); // 将输入的字节流转换成字符流
    BufferedReader buf=new BufferedReader(reader);//定义一个缓冲字符
串，用于存储文档中的字符
    FileWriter writer = new FileWriter(writeName);
    BufferedWriter out = new BufferedWriter(writer);
    FileWriter writer1 = new FileWriter(writeAll);
    BufferedWriter outAll = new BufferedWriter(writer1);
    while((s= buf.readLine())!=null){
        out.write(((double)variance(s)/15+"    "+CharNum(s)+"\r\n");
    }
    buf.close();                                //读完后关闭数据流
    out.close();
    outAll.close();
}
catch(Exception e){
    e.printStackTrace();
}
}
// 不同字母个数
public static int CharNum(String s){
    int k = 0;
    int count = 0;
    String str = "";
    try{
    int len = s.length();
    char[] c = new char[len];
    for(int i=0;i<len;i++){
        c[i] = s.charAt(i);
    }
    for(int i=0;i<len;i++){
        k=i+1;
        while(k<len-count){
            if(c[i]==c[k]){
                for(int j=k;j<len-1;j++){
                    c[j] = c[j+1];//出现重复字母，从 k 位置开始将数组往前挪位
                }
                count++;//重复字母出现的次数
                k--;
            }
            k++;
        }
    }
    for(int i=0;i<len-count;i++){

```

```

        str+=String.valueOf(c[i]);
    }
}
catch(Exception e){
    e.printStackTrace();
}
return str.length();
}
//字母频率
public static String Average(String s){
    ArrayList<Double> arr= new ArrayList<Double>();
    String str="abcdefghijklmnopqrst";
    double[]
p={0.0201,0.0324,0.0513,0.0508,0.0455,0.0658,0.0459,0.0511,0.0614,0.0771,0.0751,0.0
614,0.0537,0.0490,0.0449,0.0469,0.0524,0.0589,0.0407,0.0298};
    for(int i=0;i<s.length();i++){
        char ch=s.charAt(i);
        int sta=str.indexOf(String.valueOf(ch));
        double d=p[sta];
        arr.add(d);
    }
    double c1=0;
    DecimalFormat df = new DecimalFormat("0.00");
    for(int j=0;j<arr.size();j++){
        c1+=arr.get(j);
    }
    double t=(double)c1/15;
    String st=df.format(t);
    return st;
}
public static double variance(String s){
    String str="abcdefghijklmnopqrst";
    double[]
p={0.0201,0.0324,0.0513,0.0508,0.0455,0.0658,0.0459,0.0511,0.0614,0.0771,0.0751,0.0
614,0.0537,0.0490,0.0449,0.0469,0.0524,0.0589,0.0407,0.0298};
    double ddif=0.0;
    double sum=0.0;
    for(int i=0;i<s.length();i++){
        char ch=s.charAt(i);
        int sta=str.indexOf(String.valueOf(ch));
        double d=p[sta];
        double dif=d-Double.parseDouble(Average(s));
        ddif=dif*dif;
        sum+=ddif;
    }
    return ddif;
}
}

```

6、生成 Gauss 分布概率（Matlab）

```
close all; clear all; clc
x=-5:1:5; %x 的区间为[-5:5]，步长为 0.1
norm=normpdf(x,0,0.5); %norm 储存生成的概率
```

7、减法聚类 (Matlab)

```
x=xlsread('data3.xlsx','Sheet1');
[C,S]=subclust(x,[0.5 0.5 0.5],[],[1.5 0.5 0.15 0])
plot(0,0);
hold on;
plot3(C(:,1),C(:,2),C(:,3),'r+','markersize',16,'LineWidth',3);
hold on;
plot3(x(:,1),x(:,2),x(:,3),'g*');
view(3);
xlabel('','Interpreter','latex','String','$\bar{p}_w$', 'FontSize',15);
ylabel('k','FontSize',15);
zlabel('','Interpreter','latex','String','$p_s$', 'FontSize',15)
grid on;
```

8、汉明距离(Java)

```
import java.io.*; //导入 io 类包
import java.util.*; //导入 util 类包
public class ham {
    public static void main(String[] args) {
        real(args[0],args[1]);
    }
    public static void real(String s1,String s){
        File file=new File(s1);
        File writeName = new File(s);
        ArrayList<String> arr=new ArrayList<String>();
        String line=null;
        ArrayList<Integer> sum=new ArrayList<Integer>();
        try{
            writeName.createNewFile(); // 创建新文件,有同名的文件的话直接覆

            FileWriter writer = new FileWriter(writeName);
            BufferedWriter out = new BufferedWriter(writer);
            BufferedReader buf = new BufferedReader(new FileReader(file));
            while((line = buf.readLine())!=null){
                arr.add(line);
            }
            String str1=null;
            str1=arr.get(0);
            for(int j=1;j<arr.size();j++){
                String str2=null;
                str2=arr.get(j);
                //float similarity=hamdis(str1,str2);
                //System.out.println(str1+" "+str2+" "+similarity+"\r\n");
                out.write(str1+" "+str2+" "+hamdis(str1,str2)+"\r\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

        out.flush();
    }
    out.write("\r\n");
    out.close();
    System.out.println("写入完成！");
}
catch(Exception e){
    e.printStackTrace();
}
}
public static int hamdis(String str1, String str2) {
    int distance;
    if (str1.length() != str2.length()) {
        distance = -1;
    } else {
        distance = 0;
        for (int i = 0; i < str1.length(); i++) {
            if (str1.charAt(i) != str2.charAt(i)) {
                distance++;
            }
        }
    }
    return distance;
}
public static int getWeight(int i) {
    int n;
    for (n = 0; i > 0; n++) {
        i &= (i - 1);
    }
    return n;
}
}

```

9、编辑距离(Java)

```

import java.io.*; //导入 io 类包
import java.util.*; //导入 util 类包
public class sim {
    public static void main(String[] args) {
        //要比较的两个字符串
        real(args[0], Integer.valueOf(args[1]));
    }
    public static void real(String s, int lim) {
        File file = new File("result_无概率 .txt");
        File writeName = new File(s);
        ArrayList<String> arr = new ArrayList<String>();
        String line = null;
        ArrayList<Integer> sum = new ArrayList<Integer>();
        try {
            writeName.createNewFile(); // 创建新文件,有同名的文件的话直接覆

```


盖

```

        FileWriter writer = new FileWriter(writeName);
        BufferedWriter out = new BufferedWriter(writer);
        BufferedReader buf = new BufferedReader(new FileReader(file));
        while((line = buf.readLine())!=null){
            arr.add(line);
        }
        for(int i=0;i<lim;i++){
            out.write(i);
            String str1=null;
            str1=arr.get(i);
            int count=0;
            for(int j=lim;j<arr.size();j++){
                String str2=null;
                str2=arr.get(j);
                float similarity=levenshtein(str1,str2);
                if(similarity<0.75 && similarity>0.7){
                    count++;
                    //System.out.println(str1+" "+str2+" "+similarity+"\r\n");
                    out.write("[ "+count+" ]"+str1+" "+str2+"
"+similarity+"\r\n");
                    out.flush();
                }
            }
            sum.add(count);
            out.write("\r\n");
        }
        out.close();
        System.out.println(sum);
        System.out.println("写入完成！ ");
    }
    catch(Exception e){
        e.printStackTrace();
    }
}

public static float levenshtein(String str1,String str2) {
    //计算两个字符串的长度。
    int len1 = str1.length();
    int len2 = str2.length();
    //建立上面说的数组，比字符长度大一个空间
    int[][] dif = new int[len1 + 1][len2 + 1];
    //赋初值，步骤 B。
    for (int a = 0; a <= len1; a++) {
        dif[a][0] = a;
    }
    for (int a = 0; a <= len2; a++) {
        dif[0][a] = a;
    }
    //计算两个字符是否一样，计算左上的值

```

```

int temp;
for (int i = 1; i <= len1; i++) {
    for (int j = 1; j <= len2; j++) {
        if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
            temp = 0;
        } else {
            temp = 1;
        }
        //取三个值中最小的
        dif[i][j] = min(dif[i - 1][j - 1] + temp, dif[i][j - 1] + 1,
            dif[i - 1][j] + 1);
    }
}
//System.out.println("字符串\""+str1+"\"与\""+str2+"\"的比较");
//取数组右下角的值，同样不同位置代表不同字符串的比较
// System.out.println("差异步骤: "+dif[len1][len2]);
//计算相似度
float similarity = 1 - (float) dif[len1][len2] / Math.max(str1.length(),
str2.length());
//System.out.println("相似度: "+similarity);
return similarity;
}
//得到最小值
private static int min(int... is) {
    int min = Integer.MAX_VALUE;
    for (int i : is) {
        if (min > i) {
            min = i;
        }
    }
    return min;
}
}
}

```

附录 1.2 问题二程序

1、片段库和正确文本(Java)

```

import java.io.*;
import java.util.*;
import java.util.Random;
public class dictionary {
    public static void getStringRandom(String s) {
        File file=new File(s);
        String line=null;
        String chars = "abcdefghijklmnopqrst";
        Double[]
= {0.3,0.6,0.6,0.7,0.75,0.8,0.85,0.9,0.9,0.95,0.95,0.95,0.9,0.85,0.8,0.75,0.7,0.6,0.4,0.3};
        String val = "";//片段字符串
        char c='\0';
    }
}

```

p

```

List<String> pd = new ArrayList<String>();//片段
List<Double> pr = new ArrayList<Double>();//概率
try{
    BufferedReader buf = new BufferedReader(new FileReader(file));
    Random rand = new Random();
    while((line = buf.readLine())!=null){
        int rm = rand.nextInt(7)+15;//生成 15-21 之间的随机数，包括
        for(int i = 0; i < rm; i++) {
            int m = (int)(Math.random() * 20);
            if (mingZhong(p[m])) {
                c=chars.charAt(m);
            }
            val +=c;
        }
        pd.add(val);//片段
        pr.add(Double.parseDouble(line));//概率
        val="";
        if(pd.size()>=50){
            break;
        }
        if(pd.size()>=50){
            break;
        }
    }
    buf.close();
    for (int m=0;m<pr.size() ;m++ ) {
        System.out.println(pd.get(m)+":"+pr.get(m));
    }
    StringBuilder sb = new    StringBuilder();
    File file_result = new File("AlienLang.txt");
    PrintWriter output = new PrintWriter(file_result);//创建写对象
    for (int k=0;k<30 ;k++ ) {
        int zishu = rand.nextInt(3001)+5000;//生成 5000—8000 之间的随
        机数，包括 8000
        while (true) {
            int ra = (int)(Math.random()*50);
            if (mingZhong(pr.get(ra))) {
                sb.append(pd.get(ra));
            }
            if      (sb.length()>=(zishu-pd.get(ra).length())      &&
sb.length()<=(zishu+pd.get(ra).length())) {
                output.println(sb.toString());
                sb.delete( 0, sb.length() );
                break;
            }
        }
    }
    output.close();

```

```

        System.out.println("结束");
    }catch(Exception e){
        e.printStackTrace();
    }
}
public static boolean mingZhong(double hr) {
    Random r = new Random();
    int a = r.nextInt(100);//随机产生[0,100)的整数，每个数字出现的概率为
1%
    if(a<(int) (hr*100)){ //前 hr*100 个数字的区间，代表的几率
        return true;
    }else{
        return false;
    }
}
public static void main(String[] args) {
    getStringRandom("pro.txt");
}
}

```

2、错误文本，同模型一

3、窗体截取(Java)

```

import java.io.*;
import java.util.*;
class interception{
    public static void main(String[] args) throws Exception{
        File file_source = new File("AlienLang_err.txt");
        File file_result = new File("result.txt");
        InputStreamReader is = new InputStreamReader(new
FileInputStream(file_source));//将输入的字节流转换成字符流
        BufferedReader br=new BufferedReader(is);//将字符流添加到缓冲流
        Map<String,Integer> map = new HashMap<String,Integer>();
        String str=null;
        try{
            while ((str = br.readLine()) != null){
                for (int i=0;i<(str.length()-15);i++) {
                    StringBuilder sb = new StringBuilder();
                    for (int j=0;j<15;j++) {
                        sb.append(str.charAt(i+j));
                    }
                    if(map.containsKey(sb.toString())){//看数组中否已有该元素
                        Integer tempInt = (Integer)map.get(sb.toString());//获
取该汉字的次数，并+1
                        tempInt += 1;
                        map.put(sb.toString(), tempInt);//将汉字及其出现次
数重新加入到 map 中，并且会覆盖相同内容的键
                    }else map.put(sb.toString(), 1);//没有该元素，则加入

```

```

    }
    }
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
PrintWriter output = new PrintWriter(file_result); // 创建写对象
List<Map.Entry<String,Integer>> list = new
ArrayList<Map.Entry<String,Integer>>(map.entrySet());
Collections.sort(list, new Comparator<Map.Entry<String,Integer>>() { //
根据 value 排序
    // 降序排序
    public int compare(Map.Entry<String,Integer> o1,
Map.Entry<String,Integer> o2) {
        double result = o2.getValue() - o1.getValue();
        if (result > 0)
            return 1;
        else if (result == 0)
            return 0;
        else
            return -1;
    }
});
Iterator<Map.Entry<String,Integer>> iter = list.iterator(); // 获取 List 集合
的迭代器, Map.Entry<K, V> 为迭代元素的类型
while(iter.hasNext()){
    Map.Entry<String,Integer> item = iter.next();
    String key = item.getKey();
    Integer value = item.getValue();
    output.println( key );
}
br.close();
output.close();
System.out.println("结束");
}
}

```

4、字母频率、坐标、生成 Gauss 分布概率值、减法聚类、汉明距离同模型一

5、迭代匹配搜索(Java)

```

import java.io.*;
import java.util.*;
class IterativeSearch {
    public static void main(String[] args) throws Exception {
        File file_source = new File("AlienLang_err.txt");
        File fs = new File("center.txt");
        InputStreamReader is = new InputStreamReader(new
FileInputStream(file_source)); // 将输入的字节流转换成字符流
        BufferedReader br = new BufferedReader(is); // 将字符流添加到缓冲流
        InputStreamReader is2 = new InputStreamReader(new

```

```

FileInputStream(fs));//将输入的字节流转换成字符流
    BufferedReader br2=new BufferedReader(is2);//将字符流添加到缓冲流
    File file_result = new File("result.txt");
    PrintWriter output = new PrintWriter(file_result);//创建写对象
    String str=null;
    String str2=null;
    try{
        int count = 0;
        while ((str = br.readLine()) != null){
            count++;
            output.println("*****");
            output.println( "["+count+"]" );
            String[] array = str.split(" ");
            while (true) {
                Integer co=0;
                String ke="";
                Map<String,Integer> map = new HashMap<String,Integer>();
                while ((str2 = br2.readLine()) != null) {
                    for (int i=0;i<(str2.length()-array[0].length()+1);i++) {
                        StringBuilder sb = new    StringBuilder();
                        for (int j=0;j<array[0].length();j++) {
                            sb.append(str2.charAt(i+j));
                        }
                        if      (sb.toString().equals(array[0])      &&
i<(str2.length()-(array[0].length()+2))) ) {
                            String sub = str.substring(i, i+array[0].length()+1);
                            if(map.containsKey(sub) ){//看数组中否已有该
元素
                                Integer tempInt = (Integer)map.get(sub);//
获取该汉字的次数，并+1
                                tempInt += 1;
                                map.put(sub, tempInt);//将汉字及其出现
次数重新加入到 map 中，并且会覆盖相同内容的键
                            }else map.put(sub, 1);//没有该元素，则加入
                        }
                    }
                }
                List<Map.Entry<String,Integer>>    list    =    new
ArrayList<Map.Entry<String,Integer>>(map.entrySet());
                Collections.sort(list,                new
Comparator<Map.Entry<String,Integer>>() { // 根据 value 排序
                    //降序排序
                    public int  compare(Map.Entry<String,Integer>  o1,
Map.Entry<String,Integer> o2) {
                        double result = o2.getValue() - o1.getValue();
                        if (result > 0)
                            return 1;
                        else if (result == 0)
                            return 0;
                        else

```

```

        return -1;
    }
});
    Iterator<Map.Entry<String,Integer>> iter =
list.iterator();//获取 List 集合的迭代器,Map.Entry<K, V>为迭代元素的类型
    int c =0;
    while(iter.hasNext()){
        c++;
        Map.Entry<String,Integer> item = iter.next();
        String key = item.getKey();
        Integer value = item.getValue();
        output.println( key + ":" + value);
        if (c==1) {
            co=value;
            ke=key;
        }
    }
    }
    output.println("\n");
    System.out.println("co"+co);
    if (co.doubleValue()<=Double.parseDouble(array[1])*30) {
        break;
    }else {
        array[0]=ke;
    }
    }
}
}catch(FileNotFoundException e){
    e.printStackTrace();
}
    br.close();
    output.close();
    System.out.println("结束");
}
}

```

附录二 运行结果（部分）

附录 2.1 问题一结果

1、片段库 1（片段长度固定为 15）

brokkqgffffjapp	0.157900316601788
gbgjssqqjjjjpfb	0.221841669358911
pipjkbbsbbqmmgff	0.221841669358911
lllggeeeeklliof	0.221841669358911
rrrfkkkbrqkichs	0.299454931271490
srqcjrrrrspdrro	0.299454931271490
lbpcibbborgkkde	0.299454931271490
milcmhhlhdalmhk	0.388372109966426
ppiddhjjjjddao	0.388372109966426
oppngggpmjollj	0.388372109966426
jrrhjfhiaqppql	0.483941449038287
idnggppkklmmftt	0.483941449038287
rrbbrkqhcccdrrh	0.483941449038287
fjrrrrrenjjohhtk	0.579383105522966
cccbroeqqkkcrj	0.579383105522966
jmppejjmmmmmlgg	0.579383105522966
ghhehmmmdiiaddm	0.666449205783600
mjoomjskichpnpi	0.666449205783600
dkqqqfffqhoigaa	0.666449205783600
bplqoteeqhrccf	0.736540280606647
eejjjcjimggmmef	0.736540280606647
kkkkbjrrmbttto	0.736540280606647
pfffbjbrnqhrsrr	0.782085387950912
kqfffnaoofrpiek	0.782085387950912
kojicccfjbdkcc	0.797884560802865
klorrrrfnnikppi	0.797884560802865
mkkkjsjjjjndhgc	0.782085387950912
clqbtjfmjssmdd	0.782085387950912
hllmlllkokkknnn	0.736540280606647
nlllffffifdbkl	0.736540280606647
gdsppssshpddci	0.736540280606647
sqsnnffldccnnoo	0.666449205783600
fldiirggjqlhdla	0.666449205783600
hjjifoloqnfkehc	0.666449205783600
csdgkkkpgfgjhig	0.579383105522966
prrjcrlnffokkqq	0.579383105522966
gfpetcccccttjleq	0.579383105522966
ikjejmriibnnnh	0.483941449038287
hdnnpekssnddmqr	0.483941449038287
lnntemqqqklfqg	0.483941449038287
ibcfffoggggoekk	0.388372109966426
ljhlsssskdojkh	0.388372109966426

ooooddjhtaaaaas	0.388372109966426
ssssiksntkemm	0.299454931271490
henheeeceemmdg	0.299454931271490
hnttffiqllejnn	0.299454931271490
jjjtljjedkgogcq	0.221841669358911
qqmmhhqbeqpedke	0.221841669358911
efitgelgbklmpiq	0.221841669358911
qnhfppllbiiinn	0.157900316601788

2、外星文本（部分截图）

ddsbjrokkqgffffjappptfbrnqhrsrrmktfffflorp
ttrrbbrqlcccdrrrmqfffnnaaborpiekrrbbrkqhcccl
jggdhrqcfjniirrenjjogdhtkgbgjssqqjqkjjpfmh
hrrbbrkqkfcccdrhhjjifooqogfkehcsskskntkn
lrrkqrfnnikppihnttffiqllejnnfnfjrjprregajohh
lnrrrfnikppifdiirpdgjqrhrpldkqqqffdqqhoigaa
kpemriibnnnhibcfffoggggokkrrjcrlnffokkqqg
nntemqqklfqgnhfplebbiinpanboqbtnjfmqpthsmdl
ongggpmjgpccljkqfnojifrpiekcqqbtjfmjsmddbpl
croeqqkcrjsnssikskndomgdsppptpsshpddcikjic
nnjbhllmlllkokkknnmicqmhhlhralkhkcccsbreec
okkqggfetsentileghnlggieagagccklorrrfnnik

3、窗体截取及片段概率计算（降序排序）（部分截图）

hlllffffifdbkl11:0.66666
clqbtjfmjssmdd:0.6
gdspppssshpddci:0.56666
klorrrrfnnikppi:0.56666
kqfffnaoofrpiek:0.53333
idnggppkklmmftt:0.5
mjoomjskichpni:0.5
hllmlllkokkknnn:0.46666
fjrrrrenjjohhtk:0.46666
eejjjcjimggmef:0.46666
ghhehmmmdiiaddm:0.43333
lnntemqqqqklfqg:0.43333
kojicckfjbdkec:0.43333
hjji foloqn fkeh c:0.43333
jmppejjmmmmmlgg:0.4
jrrhjfhqppqllel:0.4
ikjejmriibnnnh:0.4
sqsnfffldccnnoo:0.4
hdnnpekssnddmqr:0.4
cccbroeqqkkcrj:0.36666
rrbbrkqhcccdrrh:0.36666
prrjcrlnffokkqq:0.36666
oppngggpmjoljlj:0.36666
fldiirggjqlhdla:0.36666
nffffibrrnqhrsrr:0.33333

4、字母频率（降序排序）

j 0.0770739210822539

k	0.0751944420392092
f	0.0658428877762554
l	0.0613861285278432
r	0.0589514646184250
m	0.0537408763771386
q	0.0524827969093011
c	0.0512960255894382
h	0.0510719142672323
d	0.0507866816753340
n	0.0489937910976870
p	0.0469462285629880
i	0.0468188925844619
g	0.0459377276130616
e	0.0455506262383423
o	0.0448986660282889
s	0.0406660181020827
b	0.0323840860587477
t	0.0298169927316623
a	0.0201598321202459

5、片段坐标(σ^2, k, p_s) (部分)

	A	B	C	
1	σ^2	k	p_s	
2				
3	1.31E-07	7	0.666667	
4	4.27E-08	10	0.6	
5	8.66E-06	7	0.566667	
6	1.31E-07	8	0.566667	
7	1.52E-05	10	0.533333	
8	2.72E-05	10	0.5	
9	1.31E-07	10	0.5	
10	8.07E-06	6	0.466667	
11	1.52E-05	9	0.466667	
12	2.24E-06	7	0.466667	
13	9.13E-07	7	0.433333	
14	1.12E-06	9	0.433333	
15	5.05E-06	9	0.433333	
16	5.05E-06	11	0.433333	
17	1.33E-05	6	0.4	
18	1.31E-07	8	0.4	

6、错误分析结果 (部分)

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)			
目标片段	相似片段	汉明距离	错误类型
clqbtjfmjssmdd	clqbtjfmcssmdd	1	替换错误
clqbtjfmjssmdd	kclqbtjfmjsmdd	10	在片段左端插入字母、在片段右端删失字母
clqbtjfmjssmdd	brqbtjfmjssmdd	2	替换错误
clqbtjfmjssmdd	clqbtjfmjssddo	2	在片段右端删失字母的删失错误
clqbtjfmjssmdd	lqbtjfmkssmddk	12	在片段左端删失字母的删失错误
clqbtjfmjssmdd	qbtjfmjssmddjj	15	在片段左端插入字母的插入错误
clqbtjfmjssmdd	clqbtjfmjsgsmd	4	在片段右端插入字母的插入错误

附录 2.2 问题二结果

1、片段库 2（片段长度随机在 15~21 之间）

ojddoeirnnhh	0.157900316601788
biiijjfkknfggklogffi	0.221841669358911
nnclllilqdnbdhhdhh	0.221841669358911
ggdrhkbhdkkkjmmnnl	0.221841669358911
llbbhhifgggkjrddddke	0.299454931271490
ffaiqddbeoqqnngghjs	0.299454931271490
smcndepaahijebbcpomk	0.299454931271490
kpgpppppppdqojf	0.388372109966426
hmlclldooooobihhoppd	0.388372109966426
dddcmdblmddjgofrqc	0.388372109966426
jdfbfdhbigqqfolcc	0.483941449038287
jmkdjjphtakjkjioll	0.483941449038287
lllliiqqlokspgpcelml	0.483941449038287
mcjjjjqqqqchhho	0.579383105522966
tbpimhkcklkkqjohoqqh	0.579383105522966
nnckmkkjddchggokmof	0.579383105522966
ffgmmrrsihhefmrtjfff	0.666449205783600
egmksopggppiggmmdppp	0.666449205783600
hqdjhpaphhbabqkk	0.666449205783600
lldhrjqenrrjsbrlkked	0.736540280606647
ddmmjjrrrbimnj	0.736540280606647
bbffffcseermmmfhndb	0.736540280606647
bbbqqdcbmnnndmm	0.782085387950912
hkqhhhhqqpnltoimfnff	0.782085387950912
qhjjgnerggttlgeei	0.797884560802865
pnnnleesrrcciighnttkk	0.797884560802865
kkhjffhkkllgjdd	0.782085387950912
nnlffgssaeoomjlbd	0.782085387950912
rnniqdddqqjffbbf	0.736540280606647
fghhaaaajjqmbbdeer	0.736540280606647
jsnnddcjfffjee	0.736540280606647
slclfaappnmenpd	0.666449205783600
lnqnffflccpmkkloqj	0.666449205783600
hhkkgkdjjphjqrriij	0.666449205783600
dsmqseeghhnnoiclmppr	0.579383105522966

ggbl1hhhhikkkksqir	0.579383105522966
dldqmdhkkemmmoohho	0.579383105522966
mpitqqqbhhclllp	0.483941449038287
kjogbosbbndinncljinn	0.483941449038287
qqnrqqobiobcqqddmom	0.483941449038287
ldqqqqhhhfoeeiin	0.388372109966426
kqoooiprrbbcgkkkjfff	0.388372109966426
ffppojsmkgkpfloodne	0.388372109966426
flmbmmmlssprrrrhi	0.299454931271490
iiibicdomcjogddraf	0.299454931271490
lnnjffoifqfoaaoqggr	0.299454931271490
kbifjnnppppmaggj1	0.221841669358911
lmnngcaaeniemdn	0.221841669358911
nhdgonmmmtkgiihgflf	0.221841669358911
qqngnioeorljbjgppgtt	0.157900316601788

2、外星文本（部分截图）

pmbqkknkcckkjddichgokehsfddmmmdljrrrmbimmij
 jokmoflldhrjgenrrjsnorrrkkedrnniqddqjffbbfln
 eoqndnngghjshhkkggkdjhdjhjqdrijjkhhjffkklslj
 nptpprnniqdddqqqfbbfqhjggbeergtlgeeijjmkdjppa
 sjgenrrjbrillkhmdhkqhhhqqpnltomifnffsmcndepaal
 isgicefmrfffnhdghlfmtkgiehgflhnrnniqdddqjfkbo
 noqghnnncomnojddcjggokmofogsomgppiggmmdppphqd
 mmoohofknnddcojejjjjeehmlcldoooobihojookdkhro
 rrjikkedlbeajfnnoiqlbalooogglnqrffffp1lccpcfke
 endmmffaafkqddbeoqqqnngghlsfbmhmlssrrrrhlsmqs
 jffbbfmpitqqqbqqhclllphhkjgkdjjjpjqrrijjkhhjkl
 jffoiffoaaoqggrotnjftgfqfaaoqggaahohkkgkdpj
 rhhbnpicllhnrkbfkklktqlodsdlalfaapnmenngr

3、窗体截取及片段概率计算（降序排序）（部分）

```
esrrcciighnttkk:0.76666
bbqqdcbmnnndmm:0.7
hhhqqpnltomifnf:0.66666
eesrrcciighnttk:0.6
sopggppiggmmdpp:0.56666
hhqqpnltomifnff:0.56666
dmmnjrrrmbimnj:0.53333
egmksopggppiggm:0.53333
ffgmmrssihhefmr:0.53333
mqseeghhhnnoicl:0.53333
bpimmhkcclkkqjo:0.53333
ddmmnjrrrmbimn:0.53333
mhkcclkkqjohoqq:0.5
nnckmkkjdddchgg:0.5
qseeghhhnnoiclm:0.5
bbbqqdcbmnnndm:0.5
nnnckmkkjdddchg:0.5
mmhkcclkkqjohoq:0.5
mmmjjrrrmbimnjj:0.5
qnfffp1lccpmkkl:0.5
ksopggppiggmmdp:0.5
```

4、字母频率（降序排序）

j	0.0736271265662097
h	0.0680394958384898
d	0.0651500746256765
m	0.0644914565551088
q	0.0629245829194839
f	0.0628608456868483
n	0.0601785704801007
k	0.0583408136057745
l	0.0565880397082959
p	0.0532737036112453
o	0.0505489369160740
g	0.0480472505351271
b	0.0454499583052269
i	0.0435962671227419
r	0.0423427682142420
e	0.0403031767699032
c	0.0352413782114270
s	0.0287773605349678
a	0.0219946566953307
t	0.0182235370977251

5、片段坐标(σ^2, k, p_s)（部分）

	A	B	C	D
1	σ^2	k	p_s	
2				
3	1.31E-07	7	0.666667	
4	4.27E-08	10	0.6	
5	8.66E-06	7	0.566667	
6	1.31E-07	8	0.566667	
7	1.52E-05	10	0.533333	
8	2.72E-05	10	0.5	
9	1.31E-07	10	0.5	
10	8.07E-06	6	0.466667	
11	1.52E-05	9	0.466667	
12	2.24E-06	7	0.466667	
13	9.13E-07	7	0.433333	
14	1.12E-06	9	0.433333	
15	5.05E-06	9	0.433333	
16	5.05E-06	11	0.433333	
17	1.33E-05	6	0.4	
18	1.31E-07	8	0.4	

6、错误分析结果（部分）

s1.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

目标片段	相似非中心子串	汉明距离	错误类型
ddmm mjrrrmbimnj	ddmmnjrrrmbimn--	1	替换错误
ddmmmjjrrrmbimnj	ddmmmjjrrgrmbim--	6	在片段右端插入字母的插入错误
ddmmmjjrrrmbimnj	addmmmjjrrrmbim--	9	在片段左端插入字母的插入错误
ddmmmjjrrrmbimnj	ddmmjjrrrmbimnb--	8	片段左端删失字母的删失错误
ddmmmjjrrrmbimnj	ddmmmjjrrrmbimnc--	3	片段右端删失字母的删失错误
ddmmmjjrrrmbimnj	dmnmjjrrrmbimn--	11	片段左端删失、替换字母