合肥工业大学
计算机与信息学院
实验报告

课　　程：自然语言理解

姓　　名：

学　　号：2018211958

专业班级：计算机科学与技术 18-2 班

日　　期：10 月 10 日

实验一 语料库的收集与整理

一、 研究背景

语料库是存放语言材料的仓库，基于语料库进行语言学的研究就是语料库语言学，“语料库语言学已经成为语言研究的主流。基于语料库的研究不再是计算机专家的独有领域，它正在对语言研究的许多领域产生愈来愈大的影响。”这是祝贺语料库奠基人生日而出版的论文集的开场白，由此不难看出，语料库语言学越来越重要，离我们的日常生活越来越近。

由于计算机的迅速发展，语料库技术涉及的领域越来越广，国内外纷纷创建自己的语料库。

在国内，北京语言学院现代汉语词频统计语料库(1983 年,182 万字)，目前北京语言大学正面向“一带一路”战略开展语料库研究和开发工作，1991 年中国国家语言文字工作委员会开始建立国家级大型汉语语料库，以推进汉语的词法、句法、语义和语用研究，其计划规模将达 7000 万汉字清华大学汉语歧义切分语料库(1998 年,1 亿汉字)，后来在汉语树库、篇章语料库建设等方面做了大量研发工作。

但是，语料库建设中也存在不少问题，如在语料库如何设计中，就存在不少争议，一种主张认为，应建立动态的或监督语料库：文本集的收集通常是随遇的，而不是平衡的。而另一种主张认为，应该建立相对静态的、平衡的。其实每种主张均与研究或应用目的密切相关。展开来说，有以下几点：

（1）语料库建设的规范问题语料库的规范问题:主要是对语料加工而言的。虽然在语料库的发展过程中形成了《信息处理用GB13000.1字符集汉字部件规范》、TEI（Text Encoding Initiative，文本编码倡议，1998年）、CES（Corpus Encoding Standard，语料库编码标准）及国际标准SGML(Standard Generalized Markup Language，标准通用置标语言)等一系列约束语料库的标准和规范，但是语料库中建设的规范问题依旧比较严重，存在分词的标准没有完全确定和统一，文本属性的规范未能完全成熟等问题。

（2）产权保护和国家语料库建设问题:在当今社会中，虽然人们广泛关注语料库的发展，重视国家语料库的建设，但是没有制定出台对于语料库知识产权保护的法律法规，以正式出版物为资源的语料库面临版权的问题，另外也没有将国家语料库的建设和保护上升到对于国家资源保护的高期中论文度。

（3）语料库的资源共享的问题:虽然近年来语料库资源在较大范围的共享已经成为了可能，但是在资源共享方面依然存在比较严重的问题。一方面，由于建设语料库的目的不同，收集的语料信息也不同，这给资源的共享带来了一定的限制。另一方面，许多语料库资源的共享是盈利性质的，这也限制了语料库资源的共享。

（4） 语料库加工中统计垃圾的问题:当今社会，由于计算机的普及，电子文本得到普遍使用，用于生成语料库资源的越来越多，但是随着语料库容量的不断增大，语料统计中的数据稀疏现象会越来越

严重。而在统计垃圾中可能会蕴藏着许多新的语言现象，所以应该正确的地对待统计垃圾，避免统计中的数据稀疏现象。

（5）语料库发展不平衡：随着语料库的发展，语料库在个领域发展不平衡的现象越来越严重。以中国为例，某些语料库，比如英汉双语语料库，在当今得到了迅速的发展，形成的语料库规格各异、数量众多，但是在少数民族语言方面，形成的语料库数目少，规模小。所以，我们应该着眼于全局，使得语料库可以得到全面均衡的发展。

解决这些问题和争议还需要我们不断的努力。

二、 模型方法

本工程是用 N-gram 文法进行处理，N-gram 指文本中连续出现的 n 个语词。n 元语法模型是基于(n-1)阶马尔可夫链的一种概率语言模型，通过 n 个语词出现的概率来推断语句的结构。

下面介绍 N-gram 语法的理论：N-gram 语言模型计算条件概率 P(wi|w1,w2,...,wi-1)时，假设只有前 n-1 个词语对当前词的概率有影响，于是假设 P(wi|w1,w2,...,wi-1)=P(wi|wi-n+1,...wi-1)，这样计算句子出现的概率的公式简化为：

$$P(S) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)\cdots P(w_i|w_{i-n+1},w_{i-n+2},\dots,w_{i-1})\cdots P(w_l|w_{l-n+1},w_{l-n+2},\dots,w_{l-1})$$

$$= P(w_1)\prod_{i=2}^{l} P(w_i|w_{i-n+1},w_{i-n+2},\dots,w_{i-1})$$

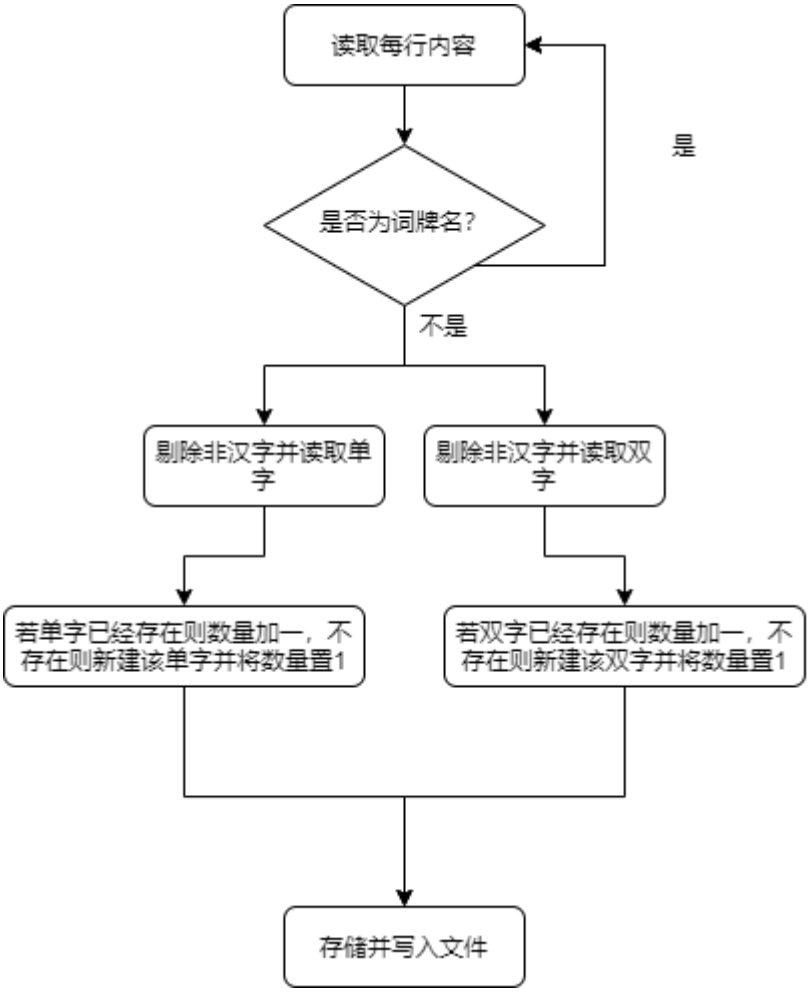
显然，上边这个式子就是一个先验概率 P(w1)乘以多个条件概率，所以也说 n-gram 语言模型是一个齐次 n-1 阶马尔可夫模型，齐次指的是状态转移概率不会随着时间的而发生变化。在实际使用中，为了

表达词语作为句首词和句尾词，我们往往会给句子加上开始标记<BOS>和结束标记<EOS>，于是上边的式子进一步得到简化：

$$P(S)=\prod_{i=1}^{l+1}P(w_i|w_{i-n+1},w_{i-n+2},\dots,w_{i-1})$$

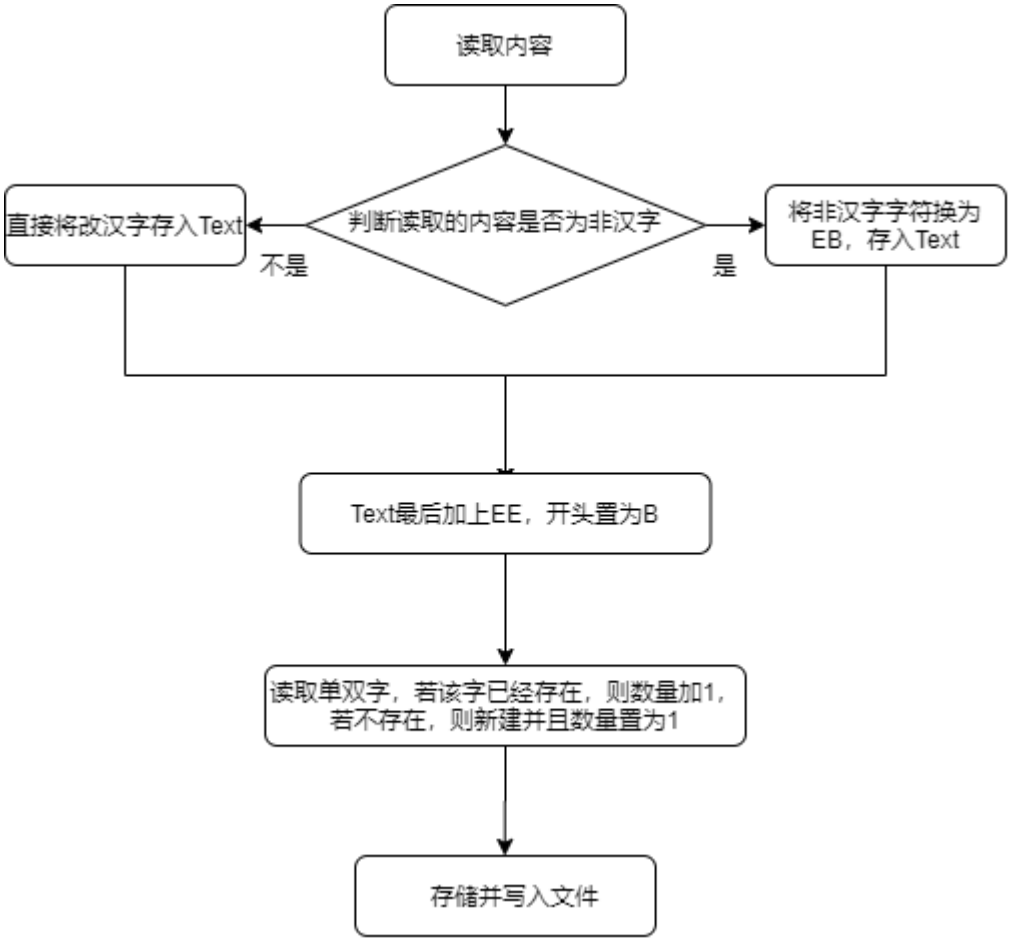
三、 系统设计

实验一我是采用 Java 来编写的，具体的核心代码设计思路可由下面的流程图来表示：



首先，我对宋词语料进行按行读取，倘若读取到的行字符长度小于 5（纵览 Ci.txt 不难发现词牌名最长为 5 字），则识别为词牌名，

跳过并继续读取下一行，若大于 5 字符，则判定为词，此时通过规定的正则表达式剔除出非汉字字符并在非字符出处分段，此后，对得到的字符串进行滑动窗口式的挨个存取，单字和双字都进行存储，若该字不存在，则新建该词并置数量为 1，若存在，则数量加 1，存储完毕后，分别写入文件 sword.txt（表示单字）和 dword.txt（表示双字）



首先，我对新闻语料库 news.txt 进行读取，读取到的汉字直接存入 Text,读取到的非汉字字符以 EB 的形式存入 Text，并将 Text 开头置 B，结尾置 EE，以使得读取的双字形式符合 N-gram 文法的定义。

最后，对得到的 Text 进行滑动窗口式的挨个存取，单字和双字都进行存储，若该字不存在，则新建该词并置数量为 1，若存在，则

数量加 1，存储完毕后，分别写入文件 sword_news.txt（表示新闻中的单字）和 dword_news.txt（表示新闻中的双字）

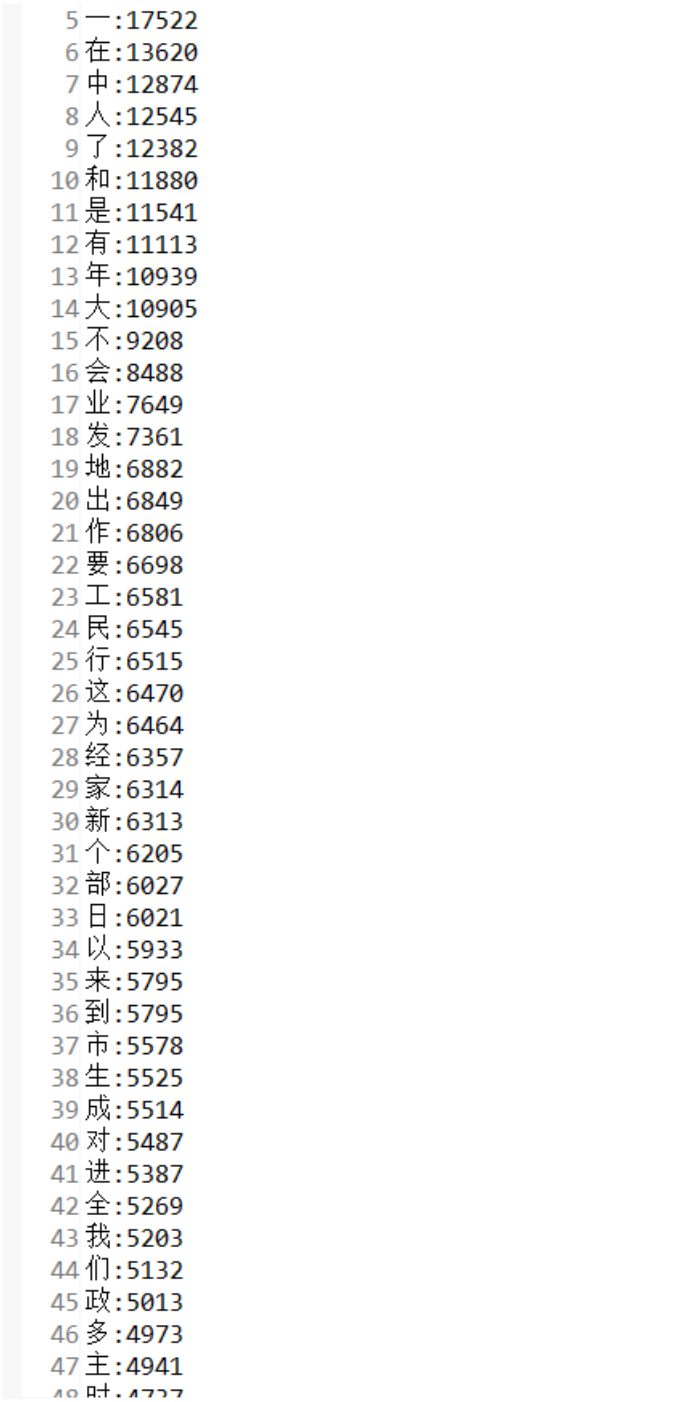
四、 系统演示与分析

运行对宋词语料库的处理程序 Experiment1_1，分别得到如下结果，第一列为单字的词和相应的词频，第二列为双字的词和相应的词频，将结果存储在 sword.txt 和 dword.txt 中。

```
1 人:13451
2 风:12875
3 花:11629
4 一:11513
5 不:10595
6 春:9963
7 无:8162
8 云:7699
9 来:7599
10 天:7517
11 月:7223
12 山:6816
13 有:6572
14 香:6505
15 时:6479
16 年:6437
17 是:5939
18 玉:5833
19 何:5730
20 如:5677
21 处:5594
22 日:5524
23 清:5510
24 相:5343
25 水:5252
26 归:5192
27 去:5154
28 红:5064
29 上:4984
30 雨:4981
31 谁:4670
32 酒:4664
33 长:4612
34 里:4519
35 金:4471
36 生:4471
37 中:4367
38 江:4351
39 今:4350
40 千:4332
41 寒:4290
42 飞:4216
43 秋:4123
```

```
1 东风:1390
2 何处:1237
3 人间:1213
4 风流:873
5 归去:831
6 春风:810
7 西风:782
8 归来:775
9 江南:768
10 相思:759
11 梅花:738
12 千里:687
13 明月:664
14 多少:658
15 回首:657
16 如今:647
17 阑干:632
18 年年:623
19 万里:595
20 一笑:592
21 黄昏:551
22 当年:546
23 天涯:538
24 相逢:536
25 芳草:532
26 一枝:518
27 尊前:518
28 风雨:508
29 流水:482
30 风吹:474
31 依旧:473
32 风月:463
33 多情:457
34 当时:457
35 故人:455
36 无人:448
37 不知:445
38 斜阳:439
39 不见:434
40 深处:425
41 时节:406
42 平生:402
43 春色:399
```

运行对宋词语料库的处理程序 Experiment1_2，分别得到如下结果，



```
一:17522
在:13620
中:12874
人:12545
了:12382
和:11880
是:11541
有:11113
年:10939
大:10905
不:9208
会:8488
业:7649
发:7361
地:6882
出:6849
作:6806
要:6698
工:6581
民:6545
行:6515
这:6470
为:6464
经:6357
家:6314
新:6313
个:6205
部:6027
日:6021
以:5933
来:5795
到:5795
市:5578
生:5525
成:5514
对:5487
进:5387
全:5269
我:5203
们:5132
政:5013
多:4973
主:4941
```

```
中国:3503
经济:3433
B在:3399
发展:3305
B这:3077
B他:2616
企业:2596
工作:2594
B我:2501
国家:2341
月日:2275
说E:2226
人民:2148
记者:2144
B一:2102
社会:2070
的E:2037
一个:2012
我们:2009
市场:1962
建设:1820
B中:1754
问题:1708
B年:1690
来E:1659
政府:1623
公司:1603
全国:1600
北京:1596
B本:1563
本报:1457
B不:1422
元E:1420
B但:1415
领导:1406
他们:1348
B有:1347
的一:1326
进行:1322
了E:1291
中央:1282
改革:1279
世界:1264
```

第一列为单字的词和相应的词频，第二列为双字的词和相应的词频，并将结果存储在 sword_news.txt 和 dword_news.txt 中。

结果都是正确的，实现了实验的要求，文件内容也都是正确而整齐有序的。

五、对本门课的感想、意见和建议

完成了实验一，我感到对于词频的处理收集统计实验是简单的，只要上课认真听了老师所说的原理，用代码实现这种程序对我们来说并不难，所以这个实验对应的课时也是合理的。

对宋词的处理在 Java 中总共是 86 行代码，对新闻的处理在 Java 中总共是 83 行代码，尽管代码量不大，但是其完成的功能却是完整的，由此可见 N-gram 文法的优越性：N-gram 文法既易于理解，又能很好的解决词频统计这类问题，其对于词频收集的思路和原理都是方便（易于实现的）而有效（结果合理）的。后面，我想尝试使用 Python，看看在 Python 各种库的加持下，能否用更短的代码来完美实现所需要的功能。毕竟设计结构方面 Java 的还是不如 Python 来的简单。

实验二、词汇知识库使用技术

一、 研究背景

从定义上看，词汇知识库是根据词义（而不是词形）组织词汇信息，从某种意义上讲，它是一部语义词典。按语义关系组织说，语义关系看作是同义词集合之间的一些指针，语义关系是双向的。如果词义{x1 ， x2 ， …} 和{y1 ， y2 ， …}之间有一种语义关系 R， 则在{y1 ， y2 ， …}和{x1 ， x2 ， …}之间也有语义关系 R。属于这两个同义词集合的单词之间的关系也是 R。

提到词汇知识库，知网就不得不提，知网是在 1988 年由董振东教授提出，它包含 4 个基本观点：

(1)NLP 系统最终需要更强大的知识库的支持。

(2)知识是一个系统，是一个包含着各种概念与概念之间的关系，以及概念的属性与属性之间的关系的系统。一个人比另外一个人有更多的知识说到底是他不仅掌握了更多的概念，尤其重要的是他掌握了更多的概念之间的关系以及概念的属性与属性之间的关系。

(3)关于知识库建设，他提出应首先建立一种可以被称为知识系统的常识性知识库。它以通用的概念为描述对象，建立并描述这些概念之间的关系，

(4)首先应由知识工程师来设计知识库的框架，并建立常识性知识库的原型。在此基础上再向专业性知识库延伸和发展。专业性知识库或称百科性知识库主要靠专业人员来完成。这里很类似于通用的词典由语言工作者编纂，百科全书则是由各专业的专家编写。

但是，知识库也存在诸多问题：

（1） 知识获取问题:由于知识的提取技术不成熟，如果采用机器自动提取知识建立知识库会降低知识库的质量，而手工建立知识库虽然保证了质量，但是效率低、成本高。因此，知识的获取是阻碍知识库发展的一个瓶颈。

（2）维护困难:由于知识的动态性，使得知识库需要经常的进行维护，当知识库规模非常大时，它的维护工作难以进行。

解决这些问题和争议还需要我们不断的努力。

二、 模型方法

本实验采取的依旧是 N-gram 语法的结果，但是引入了随机生成算法，以此来进行宋词和新闻的随机生成，关于随机生成，我们想到：

有一种基于马尔可夫链（Markov Chain）算法的随机文本生成方法，它利用任何一个现有的某种语言的文本（如一本英文小说），可以构造出由这个文本中的语言使用情况而形成的统计模型，并通过该模型生成的随机文本将具有与原文本类似的统计性质（即具有类似写作风格）。

该算法的基本原理是将输入看成是由一些互相重叠的短语构成的序列，其将每个短语分割为两个部分：一部分是由多个词构成的前缀，另一部分是只包含一个词的后缀。在生成文本时依据原文本的统计性质（即前缀确定的情况下，得到所有可能的后缀），随机地选择某前缀后面的特定后缀。在此，假设前缀长度为两个单词，则马尔可夫链（Markov Chain）随机文本生成算法如下：

设 w1 和 w2 为文本的前两个词

输出 w1 和 w2

循环：

随机地选出 w3，它是原文本中 w1w2 为前缀的后缀中的一个

输出 w3

w1 = w2

w2 = w3

重复循环

作为两个独立的实验，我们将 N-gram 语法再次介绍，N-gram 指文本中连续出现的 n 个语词。n 元语法模型是基于(n-1)阶马尔可夫链的一种概率语言模型，通过 n 个语词出现的概率来推断语句的结构。

下面介绍 N-gram 语法的理论：N-gram 语言模型计算条件概率 P(wi|w1,w2,...,wi-1)时，假设只有前 n-1 个词语对当前词的概率有影响，于是假设 P(wi|w1,w2,...,wi-1)=P(wi|wi-n+1,...wi-1)，这样计算句子出现的概率的公式简化为：

$$P(S) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)\cdots P(w_i|w_{i-n+1},w_{i-n+2},\dots,w_{i-1})\cdots P(w_l|w_{l-n+1},w_{l-n+2},\dots,w_{l-1})$$

$$= P(w_1)\prod_{i=2}^{l} P(w_i|w_{i-n+1},w_{i-n+2},\dots,w_{i-1})$$

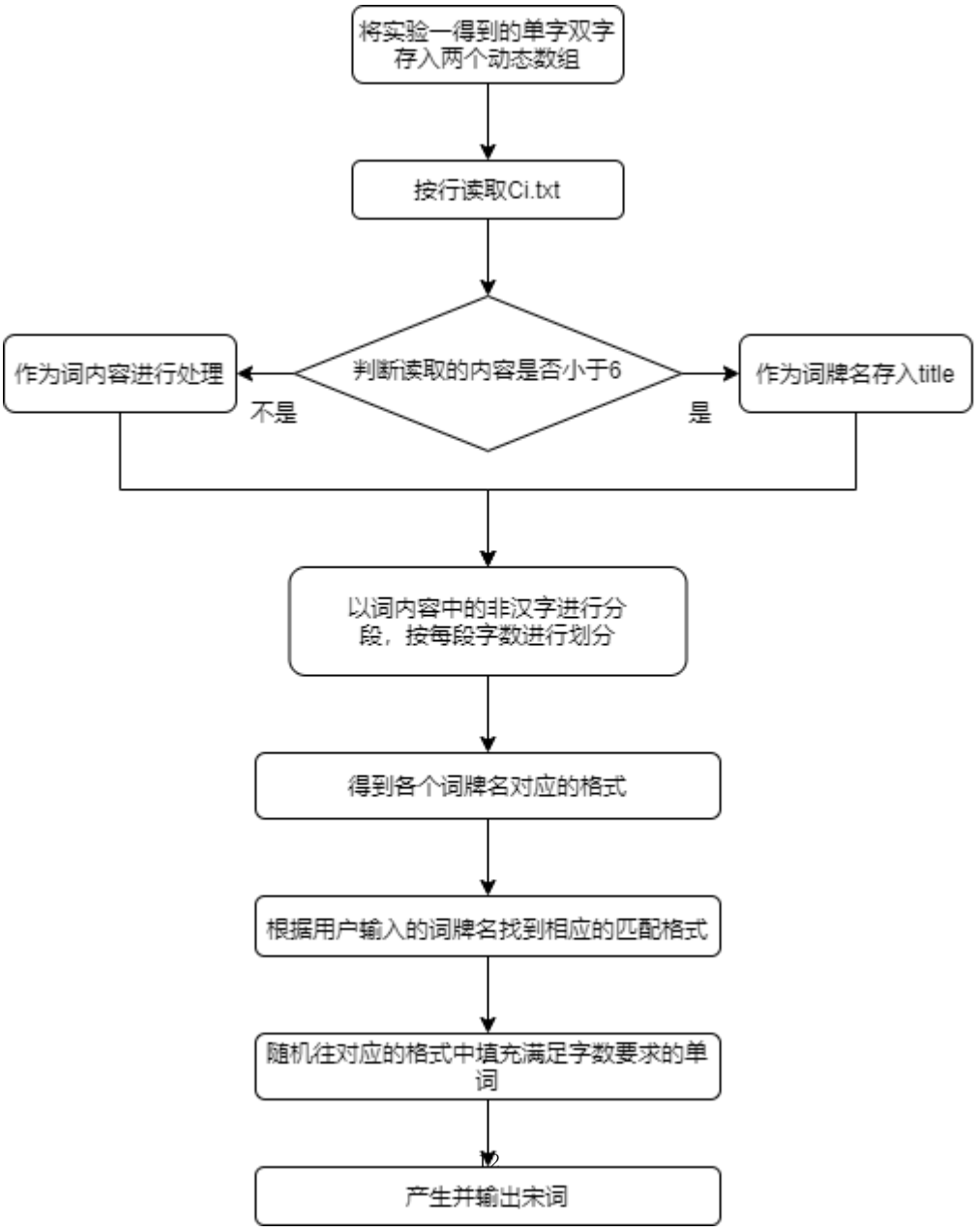
显然，上边这个式子就是一个先验概率 P(w1)乘以多个条件概率，所以也说 n-gram 语言模型是一个齐次 n-1 阶马尔可夫模型，齐次指的是状态转移概率不会随着时间的而发生变化。在实际使用中，为了

表达词语作为句首词和句尾词，我们往往会给句子加上开始标记<BOS>和结束标记<EOS>，于是上边的式子进一步得到简化：

$$P(S)=\prod_{i=1}^{l+1}P(w_i|w_{i-n+1},w_{i-n+2},\dots,w_{i-1})$$

三、 系统设计

实验二我是采用 Java 来编写的，具体的核心代码设计思路可由下面的流程图来表示：



首先，我们将实验一得到的单字双字存入两个动态数组，并按行读取 Ci.txt，并根据字数是否小于 6 判断其是否为词牌名，倘若小于 6，则是词牌名，我们继续读取下一行，倘若不小于 6，则读取到的不是词牌名而是词内容，我们将读到的词以非汉字部分进行划分，由此得到了各个词牌名对应的词的内容形式，此处，我想到既然我已经对所有的词牌名对应的格式进行了划分存储，那么我为何不将其进行输出呢，想到这，我选择对划分的结果进行了输出的存储，对应的存储文件为 struct.txt，此后，读取用户输入的词牌名，匹配到对应的格式，通过随机生成算法，往确定的格式中填入单双字，最后生成了随机的宋词并输出。

四、 系统演示与分析

运行程序，程序显示“请换行输入你想生成的宋词类型”，此处我们输入“酒泉子”，那么将自动匹配到如下的宋词格式：

```
1 酒泉子:22，223。223。23。223。223。223。23。
```

由此，将字符 2 处填入一个双字，将字符 3 处填入一个双字加一个单字，随机填入得到如下宋词：

```
<已终止> Experiment2 [Java 应用程序] E:\JDK\bin\javaw.exe (2020年10月12日 下午11:27:46)
请换行输入你想生成的宋词类型（词牌名形式）
酒泉子
酒泉子
万里十分，
不似黄昏天相思。
深处西风看一片。
风吹为一声。
长安一点水芳草。
回首时节相相逢。
凄凉东君未憔悴。
青山月梅花。
```

此外，我们将我们输出的 struct.txt 在此处列出，也就是我们得到的所有词牌名对应的格式的存储的文件：

```
酒泉子:22，223。223。23。223。223。223。23。
苏幕遮:3，3。22，23。223。22，23。3，3。22，23。223。22，23。
渔家傲:223。223。223。3。223。223。223。223。3。223。
御街行:223。3、3。223，1  23。22，22，23。223。3、3。223。222。22，22，23。
剔银灯:222。3、2、2。22，22，222。23，3、22。222。3、22。22，22，222。23，3、22。
定风波:223。223。2□22。3。223。223。223。223。3。223。
黄莺儿:223。22，22，22。22。222。23。222，22、22。2。23，23。22，22，222。222，23。222，23。
玉女摇仙佩:22，22，222。22，22，222。23。23，22。3、22，2222。23，22，22。222，22，222。22，22，222。23。3、222。3、22，22，22。3。223。
雪梅香:3，223。23，222。223，223。3，22，22。2。3，22，22。22，222。223，223。3、3，23。
尾犯:23，22，22。22，23。3、22，3、22。22，22，23。23，222。22，23。3、22，3、22。22，223。
早梅芳:3，3。223。22，22，222。22，22，23，22，22，23。  3，222，23。22，22，222。22，22，3，22，22，23。
斗百花:222，222，222，222。22，222，222。23。22。22，222。22，222。22，222，222。
甘草子:2。22，23。23，23。223。3、22。223。23。
送征衣:3，22，22，222。3、3。2。3，22，22。3、22，3、3。3、22，23。222，3、3。222，22。2。3，22，22。3、22，3、3。3、22，23。
昼夜乐:223。3、3。222，222。223。3、22。23，23。223。3、3。222，222。223，3，22。23，23。
柳腰轻:223。3、3。22，22，222。3、22，3、22。222。3、22。22，22，222。3、22，3、22。
西江月:222，222。223。222。222，222。223。222。
倾杯乐:22，22，22。3、22，22，22。223。22，222。22，222。3、22。2222。223，222。3、22。3、22。3，3、22。
笛家弄:22，22，22，223。22，22，22，22。22，22，23。3、3，22，22。2。22，22，22，22，22，22。2、222，222。22，22，23。3，3，22，22。
迎新春:23，222。23。3、3。3、22。3、222。23。3、22。22，22。3、222。223，3、22。3、223。22。22，222。
曲玉管:22，22，223。222，22。3。22，22，223。22，222。3。 22，3、22，222，222。3。23，222，22，22，22。
满朝欢:22，22，222。222，22。22，23，22。22，22，22。222，22，222。22，222。22，22，222。222，222。
梦还京:222，23。22，22，22，222。3。22，223。3、22。3。222。3。223，22，222。
凤衔杯:223。3、22。23，2 23。3、3。3，3。3、22。23，223。3、3。
鹤冲天:22，23。23，3。23，3、3。23。22，222。223。23，3。23，3、3。23。22。222。
受恩深:23。23。223。23，223。23。23，22。223。22，222。23，223。23。23，22。
看花回:223。22。223，3、22。23，22。223。22。223，222。23，22。
柳初新:223。3，3。22。22，222。222。3、22。222。3、22。22，22，222。3、22。3、22。
两同心: 22，22。3、22，3、22。3。22，22。222。22。3、22，3、22。3，22，22。
女冠子:22。3、3。3、22。22，22，22。222。223。22，22，22。22，222，223。22，3、22，23。222。223。22，22，22。
玉楼春: 223。223。223，223。223。223。223，223。
金蕉叶:223。3、22。22，223。222。223，3、22。222，23，222。
惜春郎:223。23，22，22，22。223。23。3、22，222。
传华枝:22，22。3、23。3，3，22。3，1 2，22。3、22，22。23。223，3、23。3，3，22。222，22。3、22，3、22。
雨霖铃:22。22，22。222，3、22。222，23。3、22，223。223。3、23。222，3、22。22，22、22。3、22，23。
尉迟杯:3。2222。222，222。22。3、23。3、22，222。222。3、222。22，22，222。3、22。3、23。3、22，222。
慢卷绸:22，22，23。23，22，22，22。22，22，23。23，22，22。22。3、23。23，222，222。22，22，23。23，22，22。
征部乐:22，223。3、22。3、22。3、3。222。3、3 1，223。22，22，222。3、22，22，23。23。3，22。3、22，23。
佳人醉:222。222。23。22，22。222，23。3。22。22，222，23。3。22。222。
迷仙引:22，22，22。22，222。3、3，23。3、222，22。23，23。22，223。3、22。23，22。
归朝欢:223。223。223，223。23。223。3，22，23。223。223。223，223。23。223。3，22，23。
采莲令:3，23。3、22。223，23。3、22，22，222。 22，223。3、22，22，3，23。3、22，22，222。
秋夜月:22。3、222。3、223。3、3，222。222。22。23，23。222，22。22，23。22，22。
```

此处，我仍然使用的是 Java，通过 119 行代码实现了根据实验一得到的结果，随机生成宋词的功能。但是观察结果我发现，这种根据单词双词生成得到的宋词结果有点不通顺，实际上应该是 223 这种读法的，由此我加入了一个三字（N-gram）统计功能，加入的核心代码如下：

```
49    for(int i=2;i<LText.length();i++){
50        String str=LText.substring(i-2,i+1);
51        int t=Pattern.compile("[\u4E00-\u9FA5]*").split(str).length;
52        if(t!=0)
53            continue;
54        if(tword.containsKey(LText.substring(i-2,i+1))){
55            int tem=tword.get(LText.substring(i-2,i+1))+1;
56            tword.put(LText.substring(i-2,i+1), tem);
57        }else{
58            tword.put(LText.substring(i-2,i+1), 1);
59        }
60    }                          //统计三词次数
```

```
List<Map.Entry<String, Integer>> tlist = new ArrayList<Map.Entry<String, Integer>>(tword.entrySet());
Collections.sort(tlist, new Comparator<Map.Entry<String, Integer>>() {
    public int compare(java.util.Map.Entry<String, Integer> o1, java.util.Map.Entry<String, Integer> o2) {
        return o2.getValue().compareTo(o1.getValue());
    }
});
```

```
111             }else if(text.charAt(i)=='3'){
112                 output+=tword.get(m).split(":")[0];
```

```
45          tw=btread.readLine();
46          while(tw!=null){
47              tword.add(tw);
48              tw=btread.readLine();
49          }
```

并且统计相应的词频，得到的结果如下：

tword.txt - 记事本

文件(F) 编辑(E) 格式

倚阑干:125
知何处:107
广寒宫:94
到如今:90
东风吹:87
留不住:79
人何处:76
有谁知:74
三十六:70
西风吹:66
云深处:65
不知何:62
人间世:62
人不见:61
人千里:58
君知否:58
与谁同:57
归去来:56
不如归:55
年今日:55
春归去:54
后行吹:54
何处是:52
花流水:48
花飞絮:48
二十四:47
花深处:47
归何处:47
江南春:46
向尊前:46
知多少:45
记当年:44
长安道:44
雨初晴:43
倚东风:43

由此生成的宋词结果要通顺许多，也符合宋词 223 的格式和读法。

五、 对本门课的感想、意见和建议

生成宋词的实验是充满趣味性的，看到系统自动产生了读起来还不错的宋词，内心会多多少少产生一点自豪感的，此处，我仍然使用的是 Java，通过 119 行代码实现了，根据实验一得到的结果，随机生成宋词的功能。此实验在我看来实际是考验了对文件存储和随机生成的理解，以及选用什么样的格式来存储相应的词牌名对应词格式的问题，我查阅了相关资料，得到了宋词的读法和分割思路，由此也对老师原来布置的单双词生成宋词产生了一些别的想法，如果我加入 3 字的词的统计会怎么样呢？这样的想法，我对一二实验进行了加强，并且存储在另外一个 Workspace 里面（毕竟要验收的还是单双字），由此，我实现了 3 字的统计，并且在随机生成宋词时候，考虑并且加入了 3 字的情况，这样，得到的宋词更加通顺了，对我的代码能力又是一次提升。

希望老师可以在以后的实验要求中提到这种思路，这样，代码方面不是很麻烦，也能让得到的宋词生成结果更加合理。

实验三、中文分词技术应用

一、 研究背景

本实验是基于实验一得到词频统计结果文件，通过 FMM 和 BMM 来对输入的句子进行分词并输出。

从 20 世纪 80 年代或更早的时候起，学者们研究了很多的分词方法，这些方法中基于词表的分词方法中，有两种方法，分别是:

1.正向最大匹配法 (forward maximum matching method, FMM)

2.逆向最大匹配法(backward maximum matching method, BMM)

虽然在部分文献和软件实现中指出，由于中文的性质，反向最大匹配法优于正向最大匹配法。在成熟的工业界应用上几乎不会直接使用 FMM、BMM 作为分词模块的实现方法。因为他们存在的问题也是明显的，以 FMM 为例：

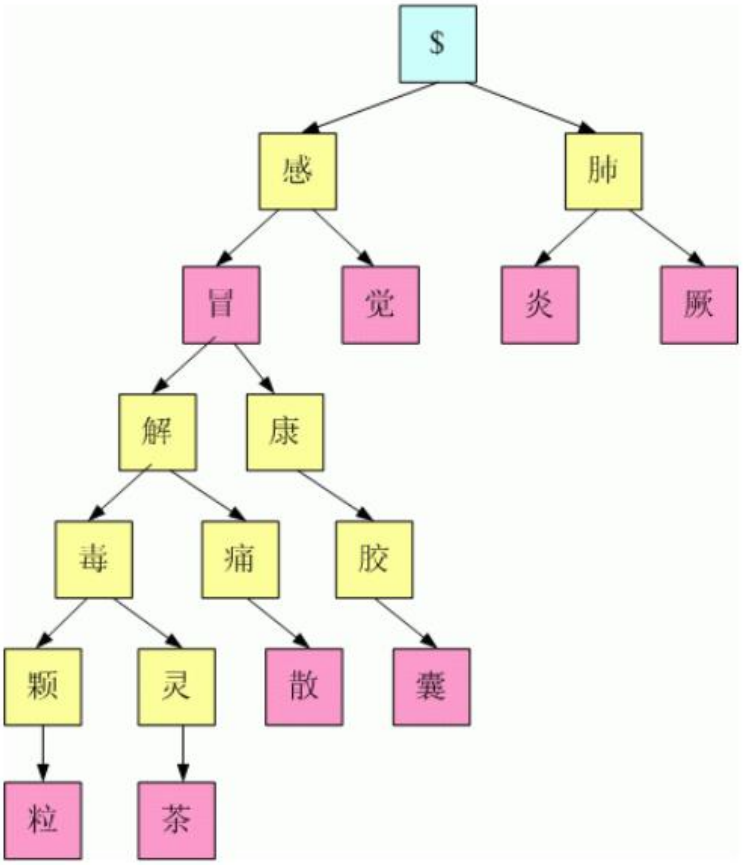
正向最大分词算法有个弊端，就是在算法开始前必须先预设一个匹配词长的初始值，而一般这个值是词典中最长词的长度，这个长度限制是最大匹配算法在效率与词长之间的一种折中。词长过长效率就比较低，词典中各个词的长度都不一致，有点较长，
但有的只是二字词或三字词。如果词长过长，在查找短字词时，将会出现许多无效的匹配，这在很大程度上影响了分词的效率。而如果初始值选取的过小，那么长词就不能得到有效的切分，达不到最大分词的目的。 根据汉语中词条的分布情况统计，在汉语中双字词语最多，而 4 字以上的词则比较少，如下表所示。可见，当初始值设置过长时，无效匹配的次数将在很大程度上消耗算法的效率。

解决这些问题仍需要我们的努力，但是通过查阅资料，我们发现，有这样一种优化算法的突破口：

真正要改善是就是我们的匹配过程，我们要减少匹配过程中的浪费，我们要解决匹配中的词长限制。但是我们有什么办法呢？每次的匹配我们必须要去词库中查找一次。怎么改善这样的做法？ 我们总是把优化的思路定格在更好的匹配算法，更好地处理词条和全文。但是真正束缚我们的却是词库！是基于关系数据库的词库，我们需要的对词库的改造，我们要让我们的词库更适合用于匹配与分词！

这是几十年来关系数据库带给我们的思维：我们查找的词是数据库的某条记录，通过表格与关系代数，我们总能找到这个词。但是正是关系数据库的这种思维束缚着我们，关系数据库让我们的数据结构及关联表达得清楚又简单，并使某些查询的效率变得很高。但是这不适用于中文分词，有的时候退到几十年前流行的数据库模型也许更适合。这就是层次数据库。 我们要做的是将关系数据库的词按字打散，并存放到层次数据库中。以下就是一个示例：

红色的字表示树上面的字串是可以单独组成一个词的，例如“感冒”它本身是词库里可以找到的词，所有红色的表示的是终止符。而黄色则表示树上面的字串是无法单独成词的，例如“感冒解”是不存在的词。真的很奇妙，词库经过这样的改装后，所有的匹配的思维都变掉了。任何一个句子都会打散成单字去与树状结构的单字去匹配，



词的长度变成了树的高度，每一次的匹配变成了树的遍历，而这种遍历的效率竟然都是线性的！

二、 模型方法

正向最大匹配法，顾名思义，对于输入的一段文本从左至右、以贪心的方式切分出当前位置上长度最大的词。正向最大匹配法是基于词典的分词方法：单词的颗粒度越大，所能表示的含义越确切，如下图所示：

0	1	2	3	4	5	6	7	8	9
我	毕	业	于	北	京	邮	电	大	学

pos	remain characters	start character	max matching
0	我毕业于北京邮电大学	我	我
1	毕业于北京邮电大学	毕	毕业
3	于北京邮电大学	于	于
4	北京邮电大学	北	北京邮电大学

反向最大匹配法的基本原理与正向最大匹配法类似，只是分词顺序变为从右至左。

0	1	2	3	4	5	6	7	8	9
我	毕	业	于	北	京	邮	电	大	学

pos	remain characters	start character	max matching
4	我毕业于北京邮电大学	北	北京邮电大学
3	我毕业于	于	于
1	我毕业	毕	毕业
0	我	我	我

容易看出，FMM 或 BMM 对于一些有歧义的词处理能力一般。举个例子：结婚的和尚未结婚的，使用 FMM 很可能分成结婚/的/和尚/未/结婚/的；为人民办公益，使用 BMM 可能会分成为人/民办/公益。

拓展一下，还存在一种双向最大匹配法：FMM 和 BMM 两种算法都分词一遍，然后根据大颗粒度词越多越好，非词典词和单字词越少越好的原则，选取其中一种分词结果输出。 如：“我们在野生动物园玩。

正向最大匹配法，最终分词结果为：“我们/在野/生动/物/园/玩”，其中，总分词数 6 个，单个词为 3。

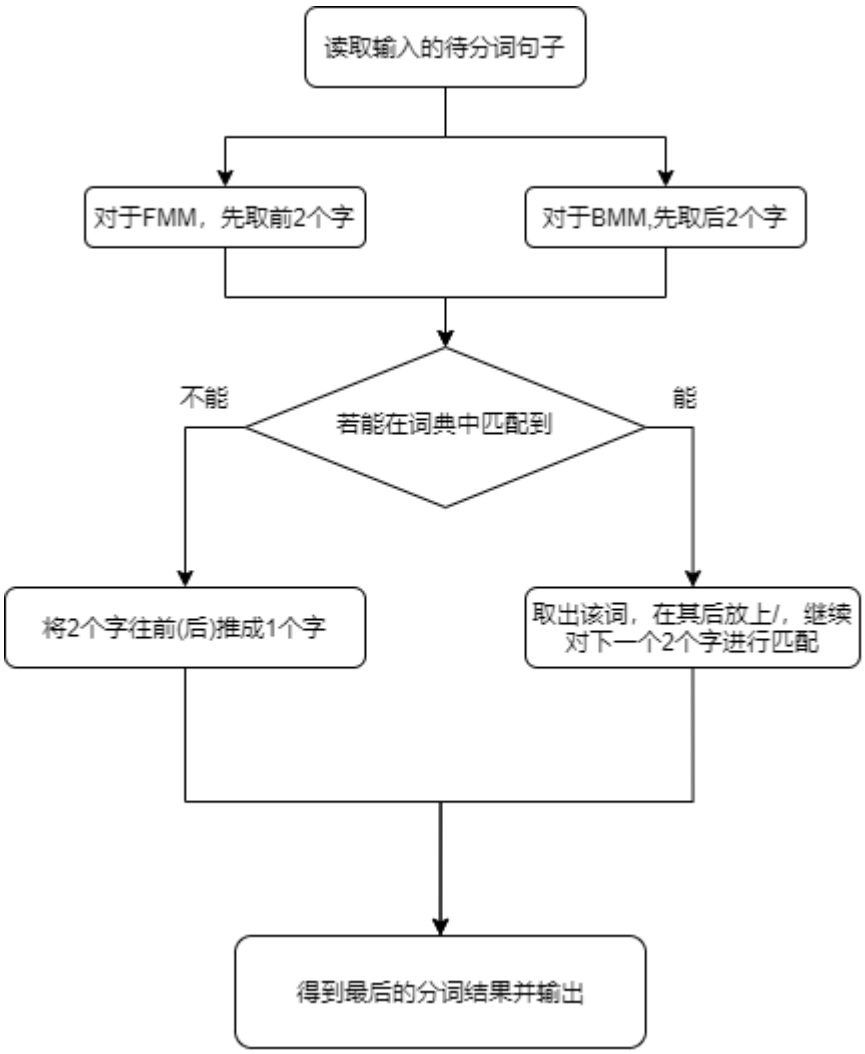
逆向最大匹配法，最终分词结果为：“我们/在/野生动物园/玩”，其中，总分词数 4 个，单个词为 2。

选择标准：首先看两种方法结果的分词数，分词数越少越好；分词数相同的情况下，看单个词的数量，越少越好；

因此最终输出为逆向结果，这就是双向最大匹配法。

三、 系统设计

实验三我是采用 Python 来编写的，具体的核心代码设计思路可由下面的流程图来表示：



首先，我们将实验一得到的新闻语料库词频分析结果读入，再读取我们的输入，由于我们实验一对新闻的分词是采用 N-gram 文法且 n=1，2，所以这里我们这里的分词结果里面划分的最长词是 2 个字的词，如果想要划分更长的段落，我们应当在处理词频时得到更长的词的词频，也就是 n 可以取得更大一点，这样对分词结果的合理度将会

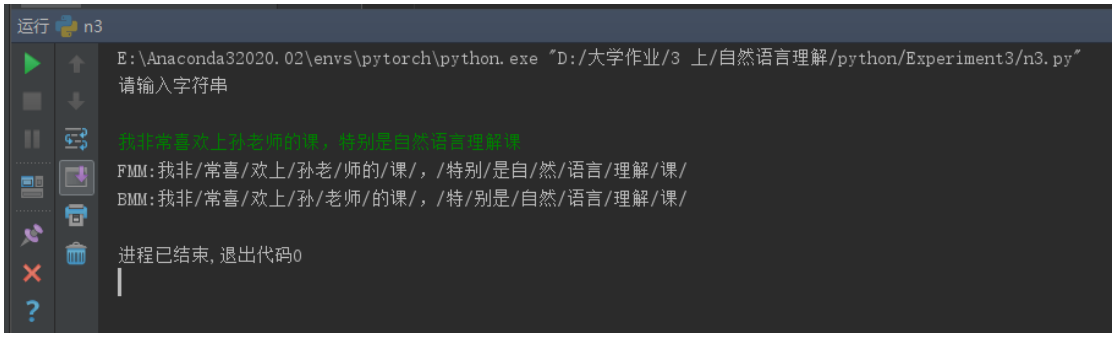
有大的提升。对于 FMM，我们先取输入的待分词句子的前两个字(也就是词典中最长的分词结果)，若该词能在词典匹配到，我们就取出该词作为结果的头部，并且在这个暂时的结果后面加上/，如果没有匹配到，我们就将搜索的长度减 1，也就是在词典中匹配输入的句子的第一个字，此时，必然能匹配到(只要新闻语料库包含的汉字够多)，BMM 与此分析过程类似，但是搜索的方向相反，就不赘述了。

四、 系统演示与分析

如图所示，我们运行该程序，运行成功，出现提示语句：“请输入字符串”，我们输入汉字字符串“我非常喜欢上孙老师的课，特别是自然语言理解课”进行测试：

得到 FMM 的分词结果为：我非/常喜/欢上/孙老/师的/课/，/特别/是自/然/语言/理解/课/

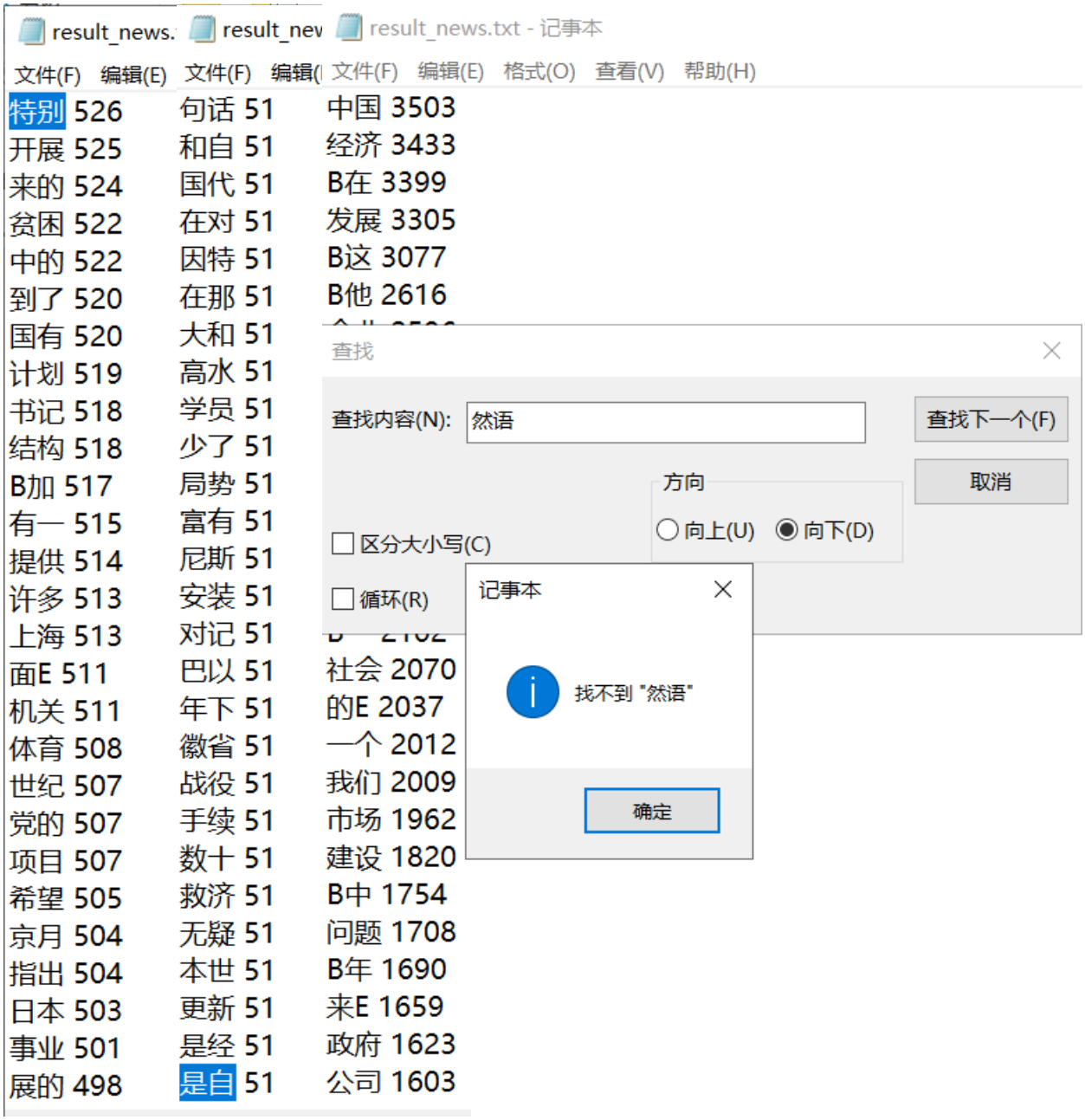
得到 BMM 的分词结果为：我非/常喜/欢上/孙/老师/的课/，/特/别是/自然/语言/理解/课/



我们不看程序，按照 FMM 和 DMM 的算法来分别分析检查一下我们得到的结果是否正确：

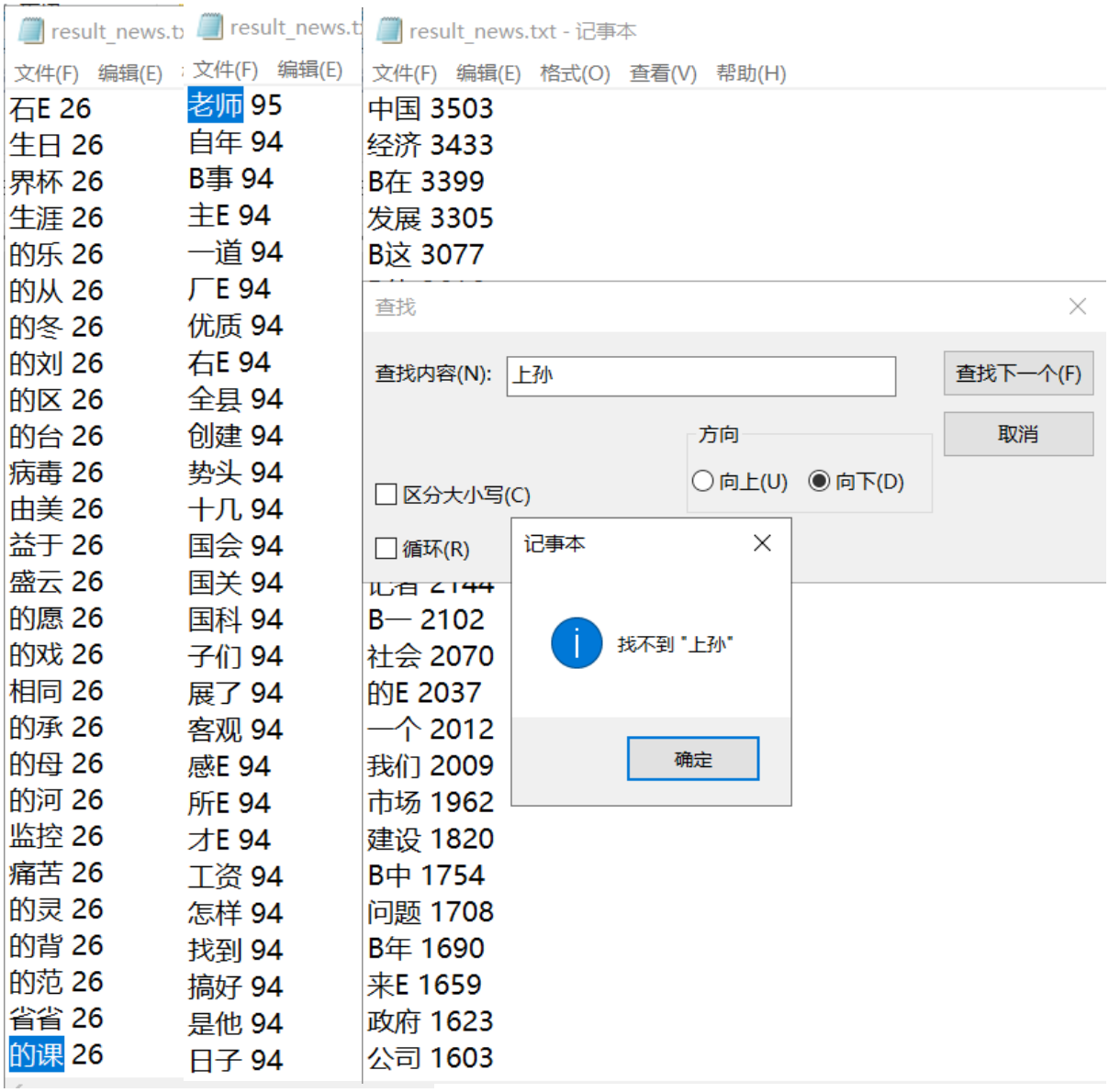
检验的思路就是挨个到新闻语料库分词的词典中去查找是否存在我们分得到的 2 个字的词的内容，如果词典中有，且结果中也匹配到了，而如果分词结果中存在单字分词结果，那么我们就去查找这个单字与接下来的一个字组成的词是否存在，如果不存在，那么就证明我们的分词结果是对的。

对于 FMM，“特别”“是自”在词典里面出现过，而“然语”却查找



不到，所以得到的 FMM 分词结果中“然”是断开的，这和我们在词典中查找到的结果是一致的。这就说明我们的 FMM 结果是对的。

接下来我们对 BMM 的分词结果进行验证：观察到 BMM 的分词结果中，存在“孙/老师/的课”，所以按照我们的验证思路，“的课”，“老师”应该是能查找到的，反之，“上孙”应该是查找不到的，否则结果就应该是“上孙/老师/的课”，我们在 result_news.txt 中查找，得到的结果是与我们预想一致的，这说明我们的 BMM 分词也是正确的。



由此，我们知道了我们代码是符合要求且正确的。

五、 对本门课的感想、意见和建议

尽管结果是正确的，但是由于我们实验一中选用的 N-gram 文法的 n=1，2，所以我们得到的分词结果看起来不是那么合理，后续我会在实验一中加大 n 的大小，然后增加一个判断语句（这是我自己的想法，我感觉这样可以滤去一些词频很小的结果，这些结果我感觉是带有干扰性的），去掉词典中那些词频很小的词（在我看来，只要处理的语料库足够大，这样的处理应该是更合理的），这样，我最后得到分词结果应该是合理通顺，符合人们正常认知的。

经过了三个简单的试验，我收获颇丰：

一是对 Java 和 Python 的优劣势比较有了一个更完整的认识，Python 在各种库，各种预置函数的加持下，代码量比 Java 会小很多。但是使用 Java 能更加锻炼我们的代码和算法能力。

二是我对各种数据结构和存取方法的掌握更加深刻了，在不同的问题上运用不同而合适的数据结构，能使得代码更高效简洁。

最后是通过这三个简单的试验，我对 N-gram 文法，FMM,BMM 分词方法了解的更加深刻了，这是在我预料之中的，毕竟“纸上得来终觉浅，绝知此事要躬行”。

实验四、文本分类技术应用

一、 研究背景

鉴于当前只要有数据集，就能用机器学习的 “万物皆可机器学习”的大背景下，我们的实验四作为一种判断文本来源进而实现分类的实验，自然是要用到机器学习的。

数字化已经改变了我们处理、分析信息的方式。网上的信息现在以指数速度增长，网页、电子邮件、期刊、电子书、学习资料、新闻和社交媒体等都充斥着大量的文字信息。为了能够快速地应对不断增长的体量，自动化处理显得越发重要。

文字分类能够智能地将不同的文章分类，它运用自动化的机器学习技术，整个过程高效简洁。近年来人工智能和机器学习发展迅速，为我们做出了极大的贡献。他们的身影无处不在，正如 Jeff Bezos（亚马逊总裁）在股东年报中指出： “在过去的几十年里，只要程序员能够清晰地写下任务处理的规则和算法，计算机就能够准确地自动执行。而机器学习的功劳在于，它使得那些不是那么清晰的任务也可以以同样的方式被计算机自动执行。”

我们肯定是使用较为简单的监督学习文本分类：如果要做监督分类，你首先要准确地定义你需要的分类结果，然后遵循『训练和测试』的原则。在训练阶段，把数据和标签传入模型，使得模型的预测结果尽量接近已有的标签。在测试阶段，把新的数据输入，观察模型的预测和真实的结果的差距，进行评价。

这种文本分类应用广泛，应用之一是垃圾邮件探测。每一封新来的邮件都会被自动分类，分类的依据是邮件的内容。语言种类的探测、情感、情绪和意图分析都属于监督学习的范围。

监督学习基本上可以看做让计算机模拟人的决策方式。计算机算法接收到一定量的标注训练数据，然后产出一个人工智能模型。这个模型会进一步被用在没有被标注的数据上，用来预测这些数据的标签，从而自动完成工作。

考虑到实验四是要求限定使用我不很熟悉的 Python 而不是我一直用的 MATLAB，我就直接使用现成的软件包——LIBSVM。

二、 模型方法

考虑到我使用的要分类的几种文本都采自搜狗新闻分类语料，其文本内容涉及的方向差别很大，用很简单的特征就能得到很好的分类效果，所以我们使用老师上课提到的两种文本特征：

（1）词频 $TF_{i,j}$，也就是特征 i 在文档 j 中出现的次数

（2）文档频率 DF_i，也就是所有文档集合中出现特征 i 的文档数

这种方法的优缺点也很明显：

优点是用的最简单的降低特征空间维数的方法，缺点是稀少的词具有着更多的信息，因此不适宜用 DF 大幅度的删除词，这也是我在实验三中考虑到的问题。

下面对 LIBSVM 进行简单的介绍:

LIBSVM 是台湾大学林智仁(Lin Chih-Jen)教授等开发设计的一个简单、易于使用和快速有效的 SVM 模式识别与回归的软件包，他不

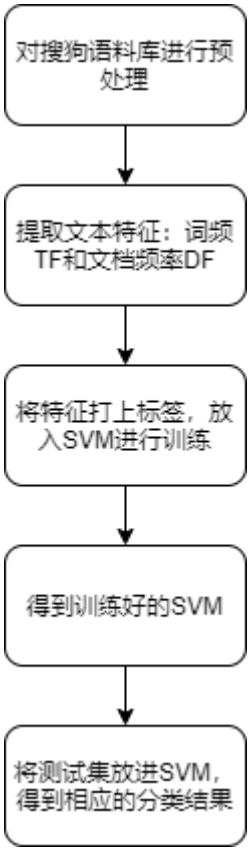
但提供了编译好的可在 Windows 系列系统的执行文件，还提供了源代码，方便改进、修改以及在其它操作系统上应用；该软件对 SVM 所涉及的参数调节相对比较少，提供了很多的默认参数，利用这些默认参数可以解决很多问题；并提供了交互检验(Cross Validation)的功能。该软件可以解决 C-SVM、 ν-SVM、 ε-SVR 和 ν-SVR 等问题，包括基于一对一算法的多类模式识别问题。

SVM 用于模式识别或回归时，SVM 方法及其参数、核函数及其参数的选择，也就是说最优 SVM 算法参数选择还只能是凭借经验、实验对比、大范围的搜寻或者利用软件包提供的交互检验功能进行寻优。

LIBSVM 拥有 C、Java、Matlab、C#、Ruby、Python、R、Perl、Common LISP、Labview、php 等数十种语言版本。最常使用的是 C、Matlab、Java 和命令行(c 语言编译的工具)的版本。

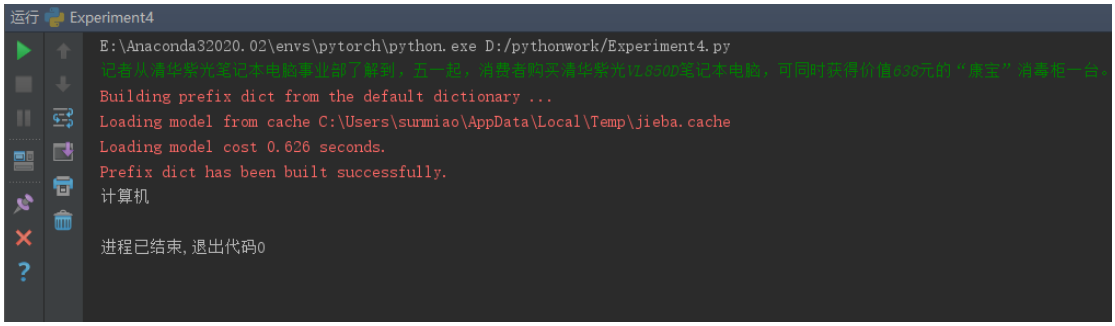
三、 系统设计

我们首先将从搜狗新闻分类语料下载的语料通过进行预处理，将得到的结果进行特征提取，进而得到一些文本特征向量，为了适应 SVM 的训练原理，我们将三种分类好的语料打上对应的标签（这里，我们在 CSDN 上下载了现有的几种分类语料，分别是：“环境”“计算机”“交通”），然后对应的打上标签 1，2，3。继而，我们将这些带标签的特征向量放入 SVM 进行训练，得到了一个训练好的 SVM，最后，我们将测试集放入 SVM 中进行分类，得到了最后的分类结果。具体的思路如下面的流程图所示：



四、 系统演示与分析

五、 我们运行程序 Experiment4，输入我们的测试集中的一句话“记者从清华紫光笔记本电脑事业部了解到，五一起，消费者购买清华紫光 VL850D 笔记本电脑，可同时获得价值 638 元的“康宝”消毒柜一台。”，这句话来自计算机类别的语料库中，运行成功，我们得到如下的运行结果——计算机，判断其来自于计算机分类语料库。



六、 对本门课的感想、意见和建议

用机器学习进行文本分类早就不是第一次做了，但是使用python进行文本训练还是我第一次做，由于对python的使用方法不是很熟悉，所以没敢尝试很复杂的方法，但是对于搜狗新闻分类语料给出的这种简单文本，我所使用的简单特征和简单模型得到的效果还是不错的，无论是验收还是自己私下测试得到的结果都是正确的。

希望老师以后可以允许我们在更多平台上进行实验，比如MATLAB等，这样我就可以做的更复杂一些，比如分更多类别，比如增加提取的特征，或者是采用深度学习来训练网络，得到结果可能也会更多一些。

下面，我来介绍用MATLAB进行文本分类的方法：

考虑到RNN这种循环神经网络对序列数据的时间步之间的长期依存关系的学习会比CNN这种适用于图像处理的神经网络更好，我们就选择了RNN中的长短期记忆网络对文本数据进行分类。（查阅资料，我们知道文本数据本身就是有序的，一段文本是一个单词序列，这些单词之间可能存在依存关系，也就是适合于RNN。）

要将文本输入到 LSTM 网络，我们首先将文本数据转换为数值序列。此处可以使用将文档映射为数值索引序列的单词编码来实现此目的。为了获得更好的结果，还要在网络中包含一个单词嵌入层。单词嵌入将词汇表中的单词映射为数值向量而不是标量索引。这些嵌入会捕获单词的语义细节，以便具有相似含义的单词具有相似的向量。它们还通过向量算术运算对单词之间的关系进行建模。例如，关系

"Rome is to Italy as Paris is to France" 通过公式 *Italy - Rome + Paris = France* 进行描述。

```
filename = "factoryReports.csv";
data = readtable(filename,'TextType','string');
head(data)
```

我们指定文本类型为‘string’，以此来将文本数据作为字符串导入。

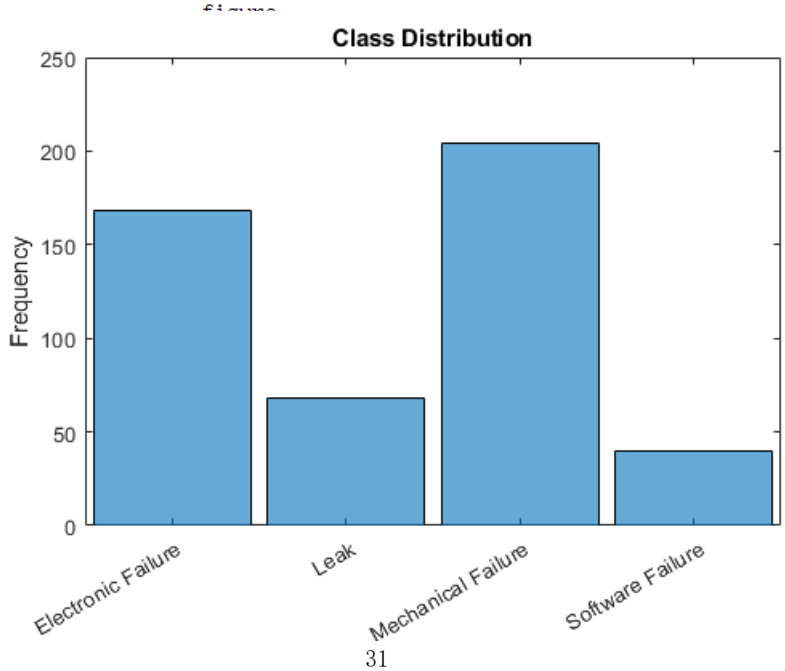
我们得到的是以下形式的结果：

Description	Category	Urgency	Resolution	Cost
"Items are occasionally getting stuck in the scanner spools."	"Mechanical Failure"	"Medium"	"Readjust Machine"	45
"Loud rattling and banging sounds are coming from assembler pistons."	"Mechanical Failure"	"Medium"	"Readjust Machine"	35
"There are cuts to the power when starting the plant."	"Electronic Failure"	"High"	"Full Replacement"	16200
"Fried capacitors in the assembler."	"Electronic Failure"	"High"	"Replace Components"	352
"Mixer tripped the fuses."	"Electronic Failure"	"Low"	"Add to Watch List"	55
"Burst pipe in the constructing agent is spraying coolant."	"Leak"	"High"	"Replace Components"	371
"A fuse is blown in the mixer."	"Electronic Failure"	"Low"	"Replace Components"	441
"Things continue to tumble off of the belt."	"Mechanical Failure"	"Low"	"Readjust Machine"	38

我们的目标是 Category 列中的标签对事件进行分类。要将数据划分到各个类，就要将这些标签转换为分类。

```
data.Category = categorical(data.Category);
```

此后，在 MATLAB 的可视化功能中，我们可以在直方图中看到数据库中类的分布。



接下来我们将其划分为训练集和验证集。按照 8：2 进行划分。并且预处理这些文本：

```
cvp = cvpartition(data.Category,'Holdout',0.2);
dataTrain = data(training(cvp),:);
dataValidation = data(test(cvp),:);
```

```
documentsTrain = preprocessText(textDataTrain);
documentsValidation = preprocessText(textDataValidation);
```

```
sequenceLength = 10;
XTrain = doc2sequence(enc,documentsTrain,'Length',sequenceLength);
XTrain(1:5)
```

由此，得到预处理后的训练数据和验证数据。

为了将文档输入到 LSTM 网络中，我们使用单词编码将文档转换为数值索引序列，然后填充和截断文档，使全部文档的长度相同

```
enc = wordEncoding(documentsTrain);
```

最后，我们使用一个 matlab 自带的 LSTM 网络架构进行训练，其中参数和训练选项都是给好的了。测试结果如下，有着 90.63%的准确度。

```
net = trainNetwork(XTrain,YTrain,layers,options);
```

Results	
Validation accuracy:	90.63%
Training finished:	Reached final iteration