自然语言导论作业报告

班级：-------- 姓名：-------- 学号：--------

题目 1

Chinese word segmentation | 中文分词

This task provides PKU data as training set and test set (e.g., you can use 80% data for model training and other 20% for testing), and you are free to use data learned or model trained from any resources.

Evaluation Metrics:
Precision = (Number of words correctly segmented)/(Number of words segmented) * 100%
Recall = (Number of words correctly segmented)/(Number of words in the reference) * 100%
F measure = 2*P*R / (P+R)

这道题目是作者的第一道题目，由于之前没有接触过 Python，所以是一边写这道题目一边学习 Python 的基本语法，而这道题里的很多语句也被作者直接用在了之后的题目里（例如正则表达式、打开文件等等）。中文分词目前主流有很多种方式，主流分为两大类，一类是基于字符串匹配，比如最大长度匹配，最少切分；另一类基于统计，引入了词性标注，例如 CRF，MMSEG；而知名的现有产品有基于 HMM 的 ICTCLAS（中科院开发），基于最大长度匹配的庖丁解牛分词，同样基于 HMM 的结巴分词（也使用到了《统计自然语言处理》中提到的引入 DAG 的算法），LTP（哈工大开发），使用 Java 编写的 ikanalyzer 等等，可以说中文分词目前方案很多。而这道题目作者使用了上课讲的最大长度匹配方式来完成，方法较为简单但是针对训练集和测试集的表现效果尚可。

最大长度匹配的基本原理是在训练过程中将训练集中的所有出现词汇写入词典，然后在识别过程中，根据设定长度，从最长的字符串开始在词典里查找，一旦匹配便加入结果中，继续匹配之后的部分；如果没有找到则长度减一，重新查找。因此最大长度匹配对词典要求比较高，如果词典中登录词较少，则表现一定会很差。针对未登录词，作者将未能查找到的单字存在列表里，一旦在其之后有词语被识别出，就将未查找到的字所在的列表全部输出。这样是基于未登录词不会连续出现的假设，但是在实际操作中，由于词典中存在单字较多，因此未登录词的单字都会被直接识别，这一点尚未想到方法去解决。最大长度匹配存在的问题不仅仅在于未登录词，还有歧义。在测试中有一个非常明显的例子，“北京大学生”这一短句在正向的最大长度匹配中得到的结果是“北京大学/生”，而在逆向的最大长度匹配中得到的结果是正确的“北京/大学生”，这一歧义问题在最大长度匹配中较难解决，在基于 CFG 或者 HMM 的算法中可能有较好的解决方案。为了试图解决这一问题，作者尝试将正向最大长度匹配和逆向最大长度匹配结合起来，先按照两种模式将一句话匹配完，然后计算两种匹配中各词语出现频次之积，借此来评判哪一种模式结果更好并加以采用。频次之积除以词典中各词语总频次即为出现概率（最大似然概率），而后者对于所有词语来说都是定值，因此直接省去。按照常理来说，“北京/大学生”出现的概率应该比“北京大学/生”出现的概率要大很多，但是由于训练集中单字的出现频次较高，因此这一方案的表现并不好，可以在下面的表格中看到

对比。

为了方便测试，作者将《1998-01-105-带音》中的词性忽略掉，同时将注音和命名实体识别所用的中括号删除，进行了一定准备操作；然后取前 18000 行作为训练集，后 4000 多行作为测试集。对于直接使用正向最大长度匹配，其最后结果为：

> Segmented:240339, Answer:231780, Right:217848
> Precision:90.6419682198894%, Recall:93.98912762101993%
> The rate of successful segment is:92.28520775482451%

即，分出 240339 个词，分词正确的有 217848 个词，测试集标准有 231780 个词，因此最后计算得到 F=92.28%。

那么将正向和逆向最大长度匹配结合起来的结果呢?

> Segmented:240225, Answer:231780, Right:218121
> Precision:90.79862628785513%, Recall:94.10691172663734%
> The rate of successful segment is:92.42317348333174%

表 1 分词的结果 F 值对比

正向最大长度匹配	92.28%
改进后的最大长度匹配	92.42%

可以看到，提升的效果不是很明显。因此如果要有明显的提升，可能需要引入 CFG 或者 HMM 等基于统计的分词方法才会更加有效。事实上基于最大长度匹配的分词对词典依赖较大，如果词典中单字较少且词语覆盖较为全面，估计 F 值表现会更好。因此如何去生成一个较好的词典对于最大长度匹配来说也是一个很重要的问题。

题目 2

N-gram Language Models | N-gram 语言模型

In this assignment you will explore a simple, typical N-gram language model.

This model can be trained and tested on sentence-segmented data of a Chinese text corpus. "Word Perplexity" is the most widely-used evaluation metric for language models.

Additional points: if you can test how does the different "Word Perplexity" of the different "N" grams, you will get additional 10 points

Additional points: if you can test how does the different "Word Perplexity" of the different smoothing methods, you will get additional 10 points

N-Gram 模型基于马尔科夫假设，一个词的出现仅与其前面 n 个词有关，而针对中文词语的 N-Gram 又被称为汉语语言模型(CLM, Chinese Language Model)。通常使用 N-Gram 模型为 bigram 和 trigram。训练 N-gram 模型使用的仍是《1998-01-105-带音》，将前 18000 个句子已经分词好的结果输入，然后用后 4000 多个句子的词语计算困惑度。在测试中发现,《1998-01-105-带音》并不是一个理想的数据源, 首先《1998-01-105-带音》中各种类文本不是随机分布的，可以看到最后有一部分诗词，对结果有一定影响；其次在试验 trigram 和 four-gram 时发现，中文中连续 3、4 个词重复出现的概率较小，与英文不同，因此使用英文语料可能结果会更好。在之后作者会尝试去通过随机抽取测试集以及使用英文语料来尝试是否能够改善模型表现。

这道题目的重点在于平滑和回退。N-Gram 模型由于存在数据稀疏和未登录词的原因，存在概率为 0 的情况。为了保证最后算得的结果不为 0，那么就要对数据进行处理。数据处理分为两方面，一是平滑，另一是回退。之前看了很长时间的古德-图灵算法，对其中的公式不是很理解，同时 Good-Turing 算法并没有考虑 r=0 和 n_r=0 的情况。之后又看了 Katz 算法, 通过将古德-图灵和回退相结合, 考虑到当出现词语的出现频次较大时，对其平滑反而会造成数据失真，因此限定平滑范围为 k<10。最开始做的+1 平滑，其效果较差，特别当应用于 trigram 和 four-gram 时，其困惑度的量级与词典大小相当，而 bigram 模型的困惑度也至少上万，说明+1 平滑对结果造成了较大的影响。当改为+δ平滑时，如果将δ设置较小（例如δ=0.0001），bigram 的困惑度将减为 3000 多，进入正常范围，但是 trigram 表现依旧不好。因此针对 trigram 作者将频次为 0 的数据其概率设置为出现在三个连续词的词典上的词语的最小概率，即用 1 除以两个连续词的词典中最大的频次，这样 trigram 的表现较为合理，但是作者个人认为这样做没有达到归一化要求。

在引入古德-图灵平滑为基础的 Katz 平滑后，即令$r^* = (r+1)\frac{n_{r+1}}{n_r}$ $(r < 10)$，且针对频次为 0 的词采用回退，即

$$p(w_i|w_{i-1},w_{i-2}) = \begin{cases} p(w_i|w_{i-2},w_{i-1}) \quad if \quad C(w_{i-2},w_{i-1},w_i) > 0 \\ \alpha_1 p(w_i|w_{i-1}) \quad if \quad C(w_{i-2},w_{i-1},w_i) = 0 \ and \ C(w_{i-1},w_i) > 0 \\ \alpha_2 p(w_i) \quad otherwise \end{cases}$$

其中$\alpha_1=\alpha_2=1$。然而得到的结果感觉并不是很科学。因此这道题目感觉需要继续考虑通过引入英文语料等其他方式完成。

困惑度的计算按照 PPT 上的方法，计算一句话中各个词语的几何平均概率后取倒数。由于概率较小，乘着乘着就小于精度变成 0 了，因此用 $\log_2$ 先取对数最后再算 2 的幂得到困惑度。

结果如下：

表 2　N-Gram 模型不同模型和不同平滑方法的结果

	add-one(δ)	**Katz**
bigram	5009.493	1491.887
trigram	1941.935	515.635
four-gram	483.934*	*

*输出每一个词语的概率发现，Four-Gram 模型里的四个连续词语基本都是未登录词，其结果完全受 add-one 中的 δ 值所左右，δ 设置较小，困惑度就小，反之，δ 设置较大，困惑度就大，因此没有参考价值。

可以看到 trigram 的困惑度小于 bigram，Katz 平滑的困惑度小于 add-one。

题目 3

Part-of-speech tagging | 词性标注

This data set contains one month of Chinese daily which are segmented and POS tagged under Peking Univ. standard.

Project ideas:

Design a sequence learning method to predicate a POS tags for each word in sentences.

Use 80% data for model training and other 20% for testing (or 5-fold cross validation to test learner's performance. So it could be interesting to separate dataset.)

目前主流的词性标注方案分为基于规则识别和基于统计两类，以统计方案为多，而常用的统计方案有基于频度、基于 N-Gram 模型和基于隐马尔科夫模型（HMM），其中最常用的就是结合 Viterbi 算法的 HMM 模型了。中文的词性标注与英文相比更为困难一点，因为英文单词本身就具有明确的词性，同时对于未登录词来说，英文拼写提供了更多信息（例如大写、前缀后缀、时态等等），而且中文相对于英文来说语法更为自由，词语的词性变化也更为丰富，因此中文词性标注的多义问题和未登录词词性识别也就变得较难解决。

这道题目作者采用了 PPT 上的方法，也就是基于 HMM 模型的 Viterbi 算法，将词性作为内在状态，而词语作为观测对象，通过将《1998-01-105-带音》的前 18000 个句子作为训练集，后 4000 多个句子作为测试集进行测试。首先是训练部分，将每一个句子中的词语和与之对应的词性提取出来，得到三个词典，分别对应三个参数 a、b 和π。a 在 HMM 模型中是从状态 i 到状态 j 的转移概率，为了运算方便，作者统计了对于一个词性 a，它转移到各词性的频次，以及它转移到其他所有词性的总频次，在测试时通过最大似然概率即可计算得 a 转移到词性 b 的概率，此为训练转移概率 a。b 在 HMM 模型中是从状态 i 发射到观测 o 的概率，即当一个词为词性 a 时，这个词是词语α的概率，也就是说 b 是从词性 a 得到词语α的概率。在训练时词典 b 统计的是每一个词性，其发射到个各词语的频次，以及该词性总发射频次。

π 是句子开始为词性 a 的概率，即 “START” 转移到词性 a 的概率。为了通过词性作为关键字进行查找，定义了一个二维词典用以存储这些信息。将三个词典训练好后，开始构建 Viterbi 算法的核心部分，即递归过程。

$$Initialization:\ V_1(j) = \pi_j \times b_j(o_1), 1 < j < N$$

$$b_1 = arg\max_{i=1}^{N} v_1(i)$$

$$Recurrence:\ v_t(j) = \max_{i=1}^{N}\left(v_{t-1}(i) \times a_{ij} \times b_j(o_t)\right), 1 \le j \le N, 1 < t \le T$$

$$b_t = arg\max_{i=1}^{N} v_1(i)$$

按照 PPT 上的算法定义了一个矩阵 viterbi[s,t]，然后对矩阵中每一项都进行计算，

对于频率为 0 的项，设置为一个较小的频率（例如 0.0001）进行平滑。然而运行以后发现效率较低，运行大约 3-4 秒钟识别完一句话，不能满足大量词语词性标注的需要。通过 debug 查看矩阵数据后发现，由于每个词语的词性不多，整个矩阵基本是稀疏矩阵，即使引入了格栅，降低了计算量，然而每一次递归仍要计算 45*45 个可能性，其中大部分为平滑的数据，将大量计算用于没有必要的地方。在郭骋城同学的启发下，将 Viterbi 矩阵降为一维，每一步仅考虑以前一步的最优解为前提得到的最优解，用局部最优解代替全局最优解，在 Viterbi 数组中仅保存最优解，用最优解去递归下一个结果，这样虽然将动态规划变成了贪心算法可能会降低准确度，但是对效率的改进是比较大的，算法复杂度从$\mathrm{m} \times n^2$降为了$\mathrm{m} \times \mathrm{n}$。而对于未登录词和未能找到其正确词性的词语，作者没有设计预测算法进行处理，直接将其标注为数字(\m)。最后的评判标准采用了最直截了当的方法，比较标注的正确率，对《1998-01-105-带音》的后 4000 多个句子进行词性标注，将标注的词性与正确词性进行比对，计算其总正确率。在训练和测试之前对《1998-01-105-带音》进行了一定处理，去掉了其中的注音和组合地名、机构名的词性标注以减少对词典的影响。最终正确率较为理想，达到了 89.53%。

题目 4

Text classification | 文本分类

This data set contains 1000 text articles posted to each of 20 online newgroups, for a total of 20,000 articles. For documentation and download, see:

http://www-2.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html.

The "label" of each article is which of the 20 newsgroups it belongs to. The newsgroups (labels) are hierarchically organized (e.g., "sports", "hockey").

You should provide model evaluation results and discuss the reasons of the results.

文本分类目前的主流算法有 Rocchio 算法、朴素贝叶斯算法、kNN 算法和 SVM 算法等等。朴素贝叶斯作为自然语言和数据挖掘中非常经典的算法，其关注的是文档属于某类别概率。文档属于某个类别的概率等于文档中每个词属于该类别的概率的综合表达式。而每个词属于该类别的概率又在一定程度上 可以用这个词在该类别训练文档中出现的次数（词频信息）来粗略估计，因此在训练阶段的主要任务就是估计各个分类的文档中出现的单词及其出现频次。在训练得到词典以后，在测试时计算测试文档中各个词语在各个类别出现的概率之积，将其标注为概率最大的类别。这样会导致有两个问题未被考虑，一是假设文本中各个词语之间是相互独立的，二是认为类别与词的出现位置无关。

$$
\begin{aligned}
c &= arg\,max_k\{P(c_k \mid x_i)\} \\
&= arg\,max_k\{\frac{P(c_k)P(x_i \mid c_k)}{P(x_i)}\} \\
&= arg\,max_k\{log\,P(c_k) + log\prod_{j=1}^{M} P(t_j \mid c_k)^{n_{ij}}\}
\end{aligned}
$$

当文本中出现未登录词时，作者将其出现的概率直接置为 2^{-100}。有理由认为，未登录词在该文档的分类过程中更倾向于负面影响而非正面影响，因为该文档的词语未在该分类出现，说明该文档不在该分类的概率更大一些。如果该词语在所有分类中均为未登录词，其对结果没有影响。

训练集采用了 “20_newsgroups”20 个新闻组各 1000 篇英文文章，共 20000 篇文章。由于不太清楚该怎么使用 Python 随机抽出文件，最后从语料库中人工随机抽取了每分类 10 篇文章作为测试集，即共有 19800 篇文章作为训练集，200 篇文章作为测试集。最终结果是在 200 篇文章中成功分类 160 篇文章，正确率在 80%。