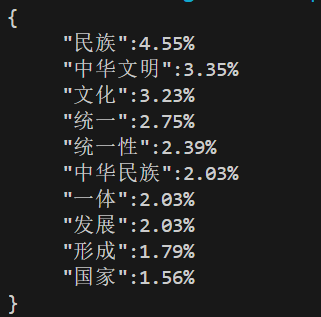
自然语言处理第一次实验报告

唐屹 10235501451

1. 实验结果截图

完整代码可在本人GitHub仓库[natural\_language\_proceing/lab1.cmain · dingzhen1145141919/natural\_language\_proceing](https://github.com/dingzhen1145141919/natural_language_proceing/blob/main/lab1.c)中查看



1. 对正相匹配算法的理解

1、时间和空间开销分析：

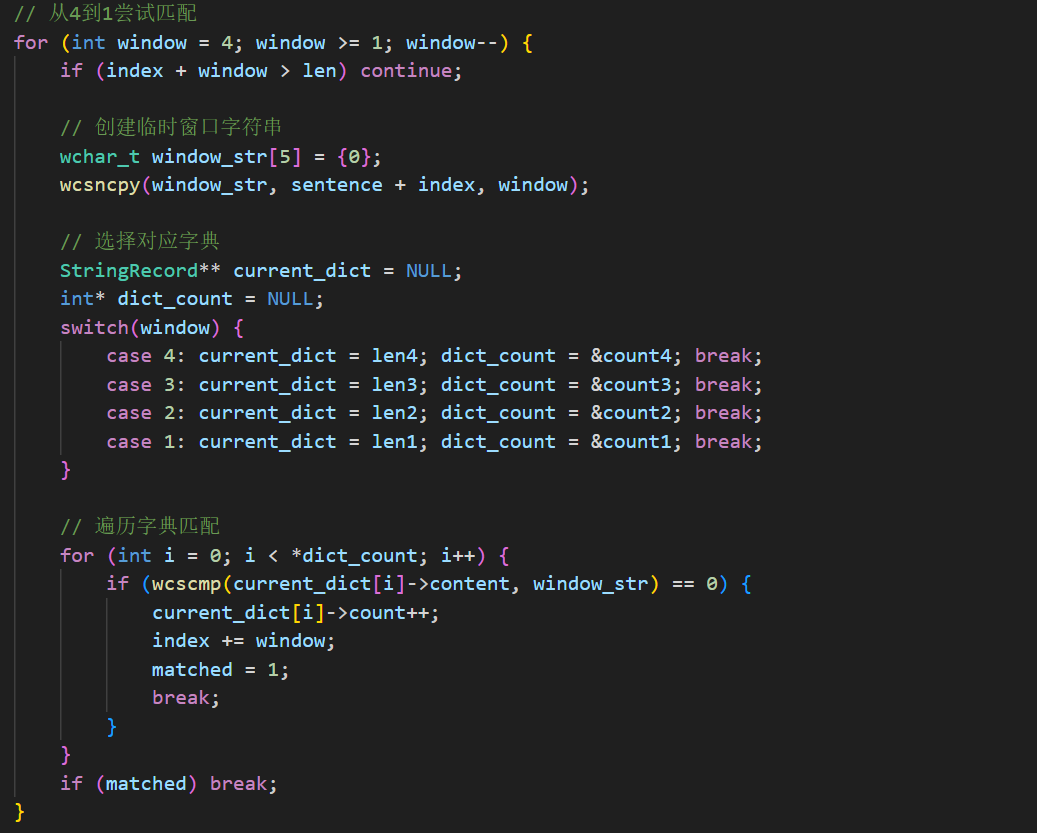
正向匹配算法是一个易于实现的算法，每次选定一个长度的字符串进行匹配，如果没有匹配到那么减少字符串长度，再进行匹配，直至顺序地对于所有可能的字符串进行匹配。因为算法思想比较简单，在开销上就会比较大，影响开销的因素也比较多。一个长度为n的句子进行分词，假设字典一共有m个单词，最坏情况下，句子的每个字都是一个词，且最大匹配长度非常大，比如为n-1，为了分出第一个单词（即句子的第一个字），需要将前n-1，n-2，……，2个字与字典的m个单词遍历比较，共n-2次循环，假设第一个字又不幸地在字典的最后一位，那么分出第一个字符就需要（n-1）\*m次比较，时间上开销非常大，因此可以尝试对于每次遍历的范围进行修改，每次遍历不需要遍历字典中的所有字词，也可以适当调整最大宽度，减少不必要的遍历，需要对于数据进行预处理；而在空间上，无论如何我们都需要开辟出一个空间用来存放字典，并且分词操作并不会对于字典的空间进行增减（换而言之，空间无法有太好的优化，因为我们必须将字典保存下来用于遍历）。

2、分词质量分析：

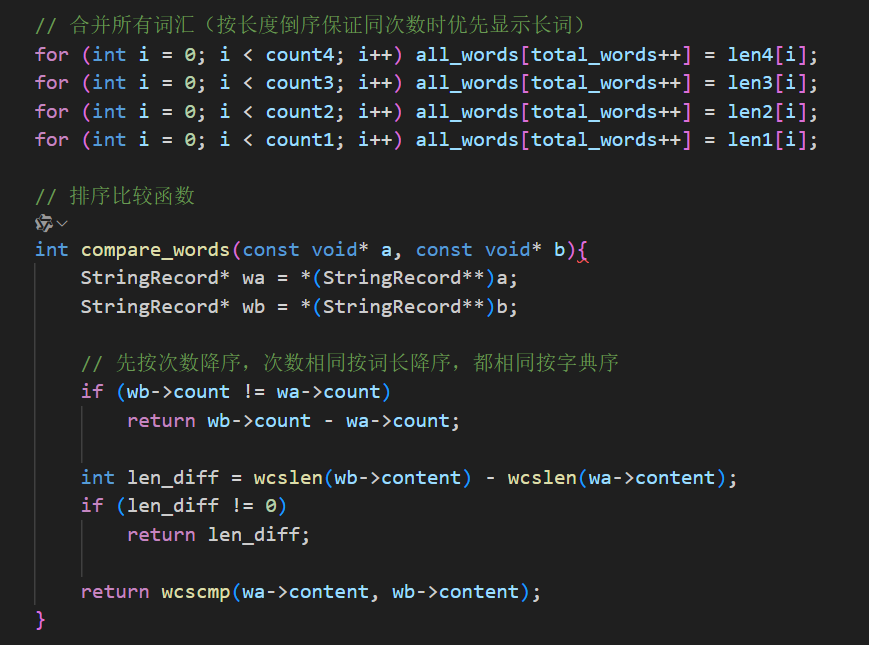
由于正相匹配算法只是单纯地对于字符串进行匹配，词典的质量、遍历顺序会影响匹配结果，比方说“上海市/长江大桥”有可能会被分为“上海/市长/江/大桥”（这个例子中，可能是因为字典中没有“上海市”这个单词），因此正相匹配算法的准确率受到词典质量的极大影响。

1. 代码实现的创新点
2. 对于词典单词的预处理

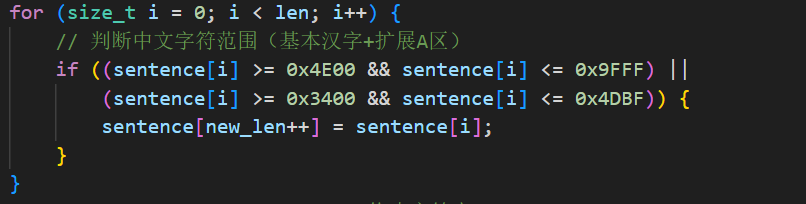
对于词典和禁用词进行数据观察之后，发现最长的单词长度为4，将单词按照不同的长度进行分组，则在匹配未知单词和字典中的单词时，无需遍历全部的字典，节省了时间。（本质上是哈希搜索，因为实验数据比较简单就直接进行了1-4的分组（对1取模），如果是对于更长的句子进行匹配哈希搜索同样能缩小遍历的范围）



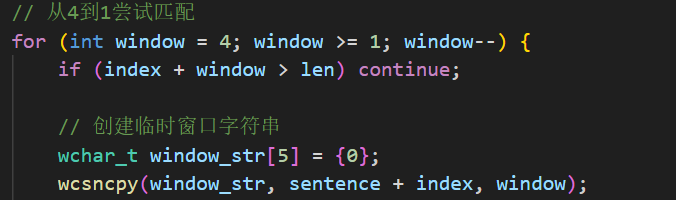
1. 输出结果更加符合实际查询情况：若频率相同，长单词在前，字典序前者在前。



1. 对于sentence文件的预处理：过滤掉标点（标点不参与统计）。



1. 优化了最大搜索长度：7->4，因为最长的单词长度就是4



1. 改进方向：

词典查找方式，可尝试树的数据结构，用单词的字典序给单词排序，提高查找效率。