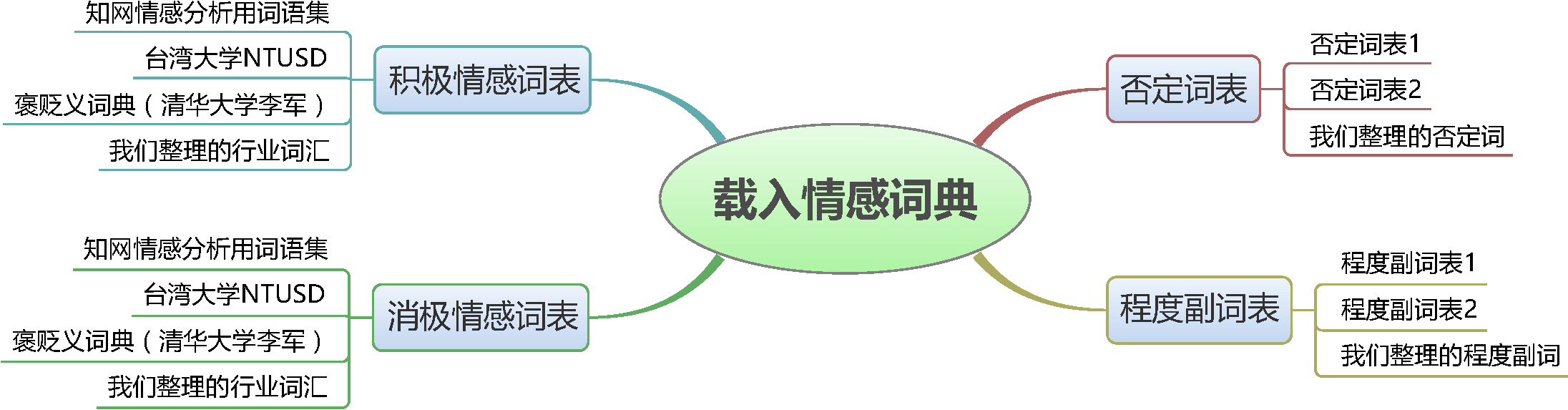


我们可以通过以下几个步骤实现基于情感词典的文本情感分类：预处理、分词、训练情感词典、判断，整个过程如上图所述。

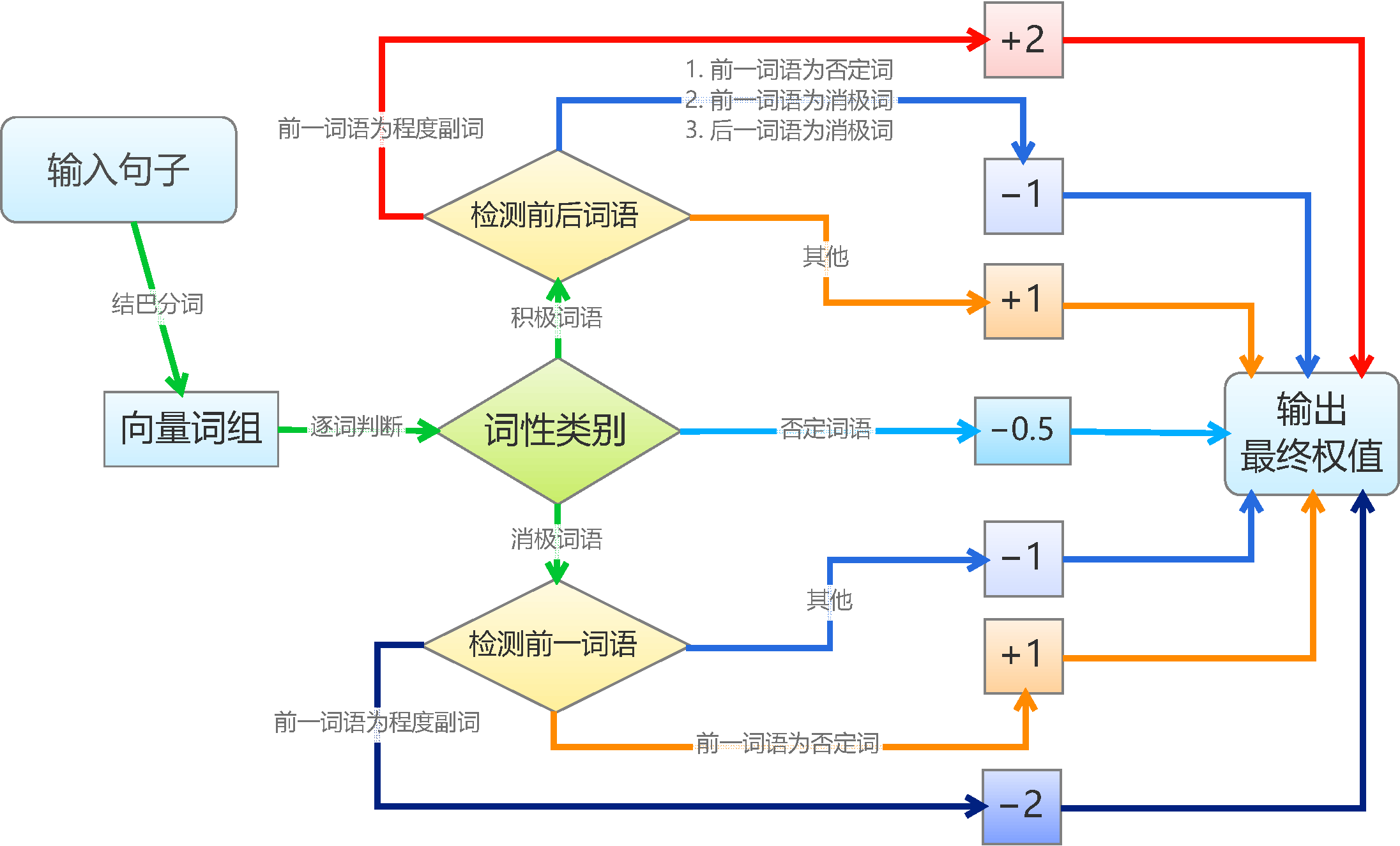
载入情感词典

一般来说，词典是文本挖掘最核心的部分，对于文本情感分类也不例外。情感词典分为四个部分：积极情感词典、消极情感词典、否定词典以及程度副词词典。为了得到更加完整的情感词典，我们从网络上收集了若干个情感词典，并且对它们进行了整合去重，同时对部分词语进行了调整，以达到尽可能高的准确率。



文本情感分类

基于情感次地啊你的文本情感分类规则比较机械化。简单期间，我们将每个积极情感词语赋予权重1，将每个消极情感赋予权重-1，并且假设情感值满足线性叠加原理；然后我们将句子进行分词，如果句子分词后的词语向量包含相应的词语，就加上向前的权重，其中，否定词和程度副词会有特殊的判别规则，否定词会导致权值反号，而程度副词则让去权重值加倍，最后，根据总权值的正负性来判断句子的情感。



要说明的是，为了编程和测试的可行性，我们作了几个假设（简化）。假设一：我们假设了所有积极词语、消极词语的权重都是相等的，这只是在简单的判断情况下成立，更精准的分类显然不成立的，比如“恨”要比“讨厌”来得严重；修正这个缺陷的方法是给每个词语赋予不同的权值，我们将在本文的第二部分探讨权值的赋予思路。假设二：我们假设了权值是线性叠加的，这在多数情况下都会成立，而在本文的第二部分中，我们会探讨非线性的引入，以增强准确性。假设三：对于否定词和程度副词的处理，我们仅仅是作了简单的取反和加倍，而事实上，各个否定词和程度副词的权值也是不一样的，比如“非常喜欢”显然比“挺喜欢”程度深，但我们对此并没有区分。

困难所在

1：语言系统是相当复杂的。

2：并不是线性的

############################第一步##################  
def yuchuli(s,m):  
 wenjian = pd.read\_csv(s, delimiter=' xxx ', encoding='utf-8', \  
 header=None, names=['comment']) # 导入文本  
 wenjian = wenjian['comment'].str.replace('(<.\*?>.\*?<.\*?>)', '').str.replace('(<.\*?>)', '') \  
 .str.replace('(@.\*?[ :])', ' ') # 替换无用字符  
 wenjian = pd.DataFrame({'comment':wenjian[wenjian!='']})  
 # wenjian.to\_csv('out\_'+s,header=False,index=False)  
 wenjian['mark'] = m #样本标记  
 return wenjian.reset\_index()  
  
neg = yuchuli('data\_neg.txt',-1)  
pos = yuchuli('data\_pos.txt',1)  
  
mydata = pd.concat([neg,pos],ignore\_index=True)[['comment','mark']] #结果文件  
#预处理基本结束  
  
  
  
############################第二步#########################  
#开始加载情感词典  
negdict = [] #消极情感词典  
posdict = [] #积极情感词典  
nodict = [] #否定词词典  
plusdict = [] #程度副词词典  
sl = pd.read\_csv('dict/neg.txt', header=None, encoding='utf-8')  
for i in range(len(sl[0])):  
 negdict.append(sl[0][i])  
sl = pd.read\_csv('dict/pos.txt', header=None, encoding='utf-8')  
for i in range(len(sl[0])):  
 posdict.append(sl[0][i])  
sl = pd.read\_csv('dict/no.txt', header=None, encoding='utf-8')  
for i in range(len(sl[0])):  
 nodict.append(sl[0][i])  
sl = pd.read\_csv('dict/plus.txt', header=None, encoding='utf-8')  
for i in range(len(sl[0])):  
 plusdict.append(sl[0][i])  
#加载情感词典结束  
  
############################第三步#########################  
#预测函数  
def predict(s, negdict, posdict, nodict, plusdict):  
 p = 0  
 sd = list(jieba.cut(s))  
 for i in range(len(sd)):  
 if sd[i] in negdict:  
 if i>0 and sd[i-1] in nodict:  
 p = p + 1  
 elif i>0 and sd[i-1] in plusdict:  
 p = p - 2  
 else: p = p - 1  
 elif sd[i] in posdict:  
 if i>0 and sd[i-1] in nodict:  
 p = p - 1  
 elif i>0 and sd[i-1] in plusdict:  
 p = p + 2  
 elif i>0 and sd[i-1] in negdict:  
 p = p - 1  
 elif i<len(sd)-1 and sd[i+1] in negdict:  
 p = p - 1  
 else: p = p + 1  
 elif sd[i] in nodict:  
 p = p - 0.5  
 return p  
#预测函数结束  
  
  
#简单的测试  
# 简单的测试  
tol = 0  
yes = 0  
mydata['result'] = 0  
for i in range(len(mydata)):  
 print(i)  
 tol = tol + 1  
 if predict(mydata.loc[i, 'comment'], negdict, posdict, nodict, plusdict) \* mydata.loc[i, 'mark'] > 0:  
 yes = yes + 1  
 mydata.loc[i, 'result'] = 1  
print(yes / tol)