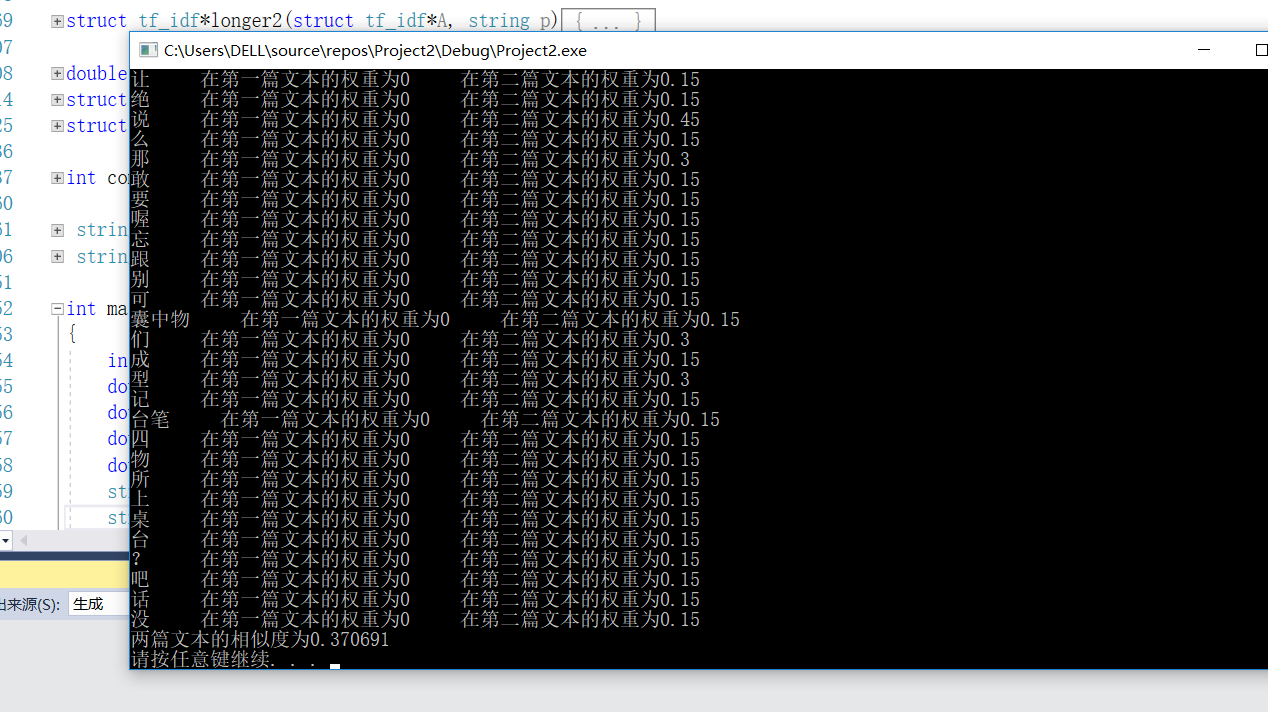
分词实验 2016211318班 王震山 学号2016211622 方昌红 学号2016211621

分词算法主要由方昌红实现

权重和相似度计算主要由王震山实现



首先给出程序跑出来的结果，代码使用的是C++，开发环境windows，具体代码比较长这里就不贴了，老师可以在压缩包Project2\Project2\源1.cpp里查看

实验要求一

使用反向最大匹配算法，词与词间我是用“/”来间隔，具体代码为

string check(string p1)

{

string p2 = "";//记录输出

string p3 = p1;//在内部计算时做中转

string test = p1;//在外部计算时做中转

int i = 0;

double num = 0;

while (1)

{

for (i = 0; i = i + 2;)

{

if ( compare(p3)==1)//比较成功

{

num++;

tt=longer(tt, p3);//

p2 = p3 +"/"+ p2;

int mark = i +2- p3.length();

test = test.substr(0, i+2-p3.length());

p3 = test;

if (mark== 0)

{

goto over;

}

break;

}

else if(i==int(test.length()))//到了最后一位

{

num++;

tt = longer(tt, p3);//

p2 = p3 + "/" + p2;

test = test.substr(0, i - 2);

p3 = test;

if (i == 2)

{

goto over;

}

break;

}

p3 = test.substr(i, test.length() - i);

}

}

over:

tt = tf1(tt, num);

return p2;

}

存入的文件位置在Project2\Project2\input.txt和input2.txt中

因为本算法在计算后面的权重和文本相似度时使用了链表，导致在每一步比较时都要遍历链表，所以分词速度比较慢，分词准确性较高

实验要求二

这段代码我写的比较松散，部分代码糅杂在了链表的建立过程中，

double tf(double n,double max)

{

double tf;

tf = n / max;

return tf;

}

struct tf\_idf\*tf1(struct tf\_idf\*A, double num)

{

struct tf\_idf\*head = A;

head = head->next;

while (head->next != NULL)

{

head->tf = tf(head->app, num);

head = head->next;

}

return A;

}这是求tf值的部分代码

实验要求三

在求出权重后这一步就很容易了，直接使用函数计算即可，求得答案见开头图片