英文非词纠错系统

English Non-Word Correcting System



小灯泡

小组成员：计算机1104 陈诒聪 20111003809

软件1102 宋惠琼 20111003524

软件1102 王文敏 20111003529

摘 要

本系统主要用于英文拼写检错与纠错，其中采用最小编辑距离法计算产生合适的可替换的真词集合以实现纠错功能。实验表明最小编辑距离法对常见单词错误的发现很有效，但自动纠错并不理想，本系统是先发现错误，一般不采用自动纠错，而是设置一个表示错词与真词词长相差多少的阈值，在一定范围内取出所有适当的真词，进行Levenshtein距离计算，再通过Levenshtein大小进行排序，最终返回给用户最优结果，用户根据需求选择想要的正确单词。本系统还增加了其他通用的功能，如单词翻译功能跟词汇的更新功能。鉴于本次通查词功能过分依赖于本地词典，而本地词典大小有限，很难满足日常查词需要，故增加单词在线翻译功能。

关键字：拼写检查、校正 单词在线、离线翻译 Levenshtein距离

Abstract

This system aims to detect and correct the spelling error of English words. It takes the method of minimum edit distance come up by Levenshtein to provide users a cluster of correct words to choose to replace the wrong one. Since the minimum edit distance can not work out the exactly right word, the system automatically detect the wrong words in a passage but replace them with the right one chosen by users from a set of words given by this system. First, the system detects wrong words in a passage. Second, it provides some words which have similar characters with the wrong one. Then the selected words will be sorted according to the their Levenshtein value and provided to users to choose to replace the wrong one. What’s more, translation and updating new words also be added as new functions.

Key words: Non-word checking and proofreading Levenshtein distance Words online/offline translation

目录

[一、 引言 4](#_Toc391554961)

[二、 国内外研究现状分析 4](#_Toc391554962)

[2.1 国外 4](#_Toc391554965)

[2.2 国内 4](#_Toc391554966)

[三、 英文拼写错误类型 5](#_Toc391554967)

[四、 英文文本中的错误发现与纠错方法 6](#_Toc391554968)

[4.1 单词错误的发现与纠错 6](#_Toc391554971)

[4.2 最小编辑距离法 7](#_Toc391554972)

[五、 最小编辑距离法 8](#_Toc391554973)

[5.1 编辑距离 8](#_Toc391554975)

[5.2 Levenshtein编辑距离 9](#_Toc391554976)

[5.3 本实验中Levenshtein距离的应用 10](#_Toc391554977)

[5.4 实际例题详细分析过程 12](#_Toc391554978)

[5.5 互联网查词功能实现 14](#_Toc391554979)

[六、 系统结果展示及分析 16](#_Toc391554980)

[七、 学习体会及对该课程的意见和建议 21](#_Toc391554981)

[八、 小组成员打分 22](#_Toc391554982)

[九、 参考文献 22](#_Toc391554983)

1. 引言

自然语言处理技术是研究和实现人与计算机之间用自然语言进行有效沟通的理论和方法。近年来,随着计算机科学技术的发展和统计学习方法的有效应用,它已经成为人工智能和语义搜索领域的重要研究方向。对于英语文本的信息处理,如果其中的语义单元(单词与句子)存在错误,则不可避免的会影响到后续文本分析和语义理解,并最终降低实际应用系统的整体性能。因此,对英语文本中的单词与语法错误进行智能检错与纠错是自然语言处理的重点和难点之一。

1. 国内外研究现状分析

## 国外

英文拼写检查是自然语言处理的主要应用领域之一。20世纪60年代，国外就开展了英文文本的自动校对研究；IBM Thomas J. Watson研究中心首先于1960年在IBM / 360和IBM / 370用UNIX实现了一个TYPO英文拼写检查器；1971年，斯坦福大学的Ralph Gorin在DEC-10机上实现了一个英文拼写检查程序Spell。多年来，随着计算机技术的不断发展，新的输入技术不断涌现，如OCR识别、语音识别，开展拼写错误校对的研究更加迫切。这方面的研究也在不断取得进展，部分成果已经商品化，目前流行的一些文字处理软件（如Word，Wordperfect等）也都嵌入了英文拼写检查功能。国际互联网上还能见到Expert Ease公司推出的Deal Proof，Newton公司推出的Proofread等英文单词拼写检查系统。

## 国内

国内在拼写校正方面的研究主要针对中文文本处理，始于20世纪90年代初期，但发展速度较快。目前有许多科技公司和高等院校或研究机构都投入了一定的人力和财力开展这方面的研究，并取得了一些较好的成果，且有部分成果已经商品化，如黑马校对系统、金山校对系统、工智校对通等。

国内在英文拼写检查方面的主要研究有：《文本自动校对技术研究综述》综述并分析了英文文本中的错误发现与纠错方法；《中文文本自动校对》中介绍了针刺错误检查的一些方法，其中提到了通过基线、上下文词和搭配来获取特征的方法；《英文作文的自动拼写检查研究》对英文作文中的拼写错误进行了分类，分别采用规则匹配和统计建模的方法对英文作文中的非词错误和真词错误进行研究；《基于上下文的真词粗偶检查及校对方法》主要研究英文文本中的真词错误检测，提取所要检测单词的上下文特征，把判断该单词使用正确与否看作分类问题，使用Winnow分类算法进行训练和测试。

1. 英文拼写错误类型

英文拼写时，常见的错误有以下几种：非词错误、真词错误和句法语义错误。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 非词错误 | 真词错误 | 句法语义错误 |
| 定义 | 非词错误是指文本中被词边界分隔出的字符串，根本就不是词典中的词，也称为“单词错误”。 | 虽不是想要的单词，但  却在词典中能够查到的真正的单词。  也称为“上下文相关的文本错误”。 | 真词错误造成  原稿本身存在语法错误  输入时丢失了某个单词甚至串行或丢失一整行。 |
| 例子 | them→tehm;  the→thr;  partition→patition;  study→studdy | from→form;  employer→employee |  |
| 原因 | 指法错误 粗心 | 指法错误 粗心 | 指法错误 粗心 客观因素 |
| 类别 | 替换错误 易位错误 丢失错误 插入错误 |  |  |
| 后果 |  | 输入的词语上下文搭配不当，不是当前语境中所需要的词。 |  |

我们主要是检测非词错误。

1. 英文文本中的错误发现与纠错方法

## 单词错误的发现与纠错

英文文本中单词错误的检测发现方法目前主要有两种，即N-gram分析法和查词典法。

N-gram错误检测技术对输入串中的每一个n元串(n一般取2或3)在实现编辑好的一个N-gram表中进行查找，看它是否在表中存在或它的出现频次，那些不存在或出现频次非常低的n元串被认为是可能拼写错误，如“shj”或“het”就是错误的三元串。N-gram分析法通常需要一个词典或大规模的文本语料以便事先编辑N-gram表。

查词典法主要是检测所输入的n元串是否在词典或可接受的此表中，如果不在词典中，则将该输入串标志为一个拼写错误的词。由于基于查词典法的校对系统差错精度高，因此是目前较为流行的错误检测技术。考虑到存取速度，当词典规模较大时，为了提高查找速度，有效的词典查找算法也是需要考虑。

单词错误的纠错方法主要有误拼词典法、词形距离法、最小编辑距离法、相似键法、骨架键法、N-gram法、基于规则的技术、词典及神经网络技术。

|  |  |
| --- | --- |
| 类型 | 介绍 |
| 误拼词典法 | 收集大规模真实文本中拼写出错的英文单词并给出相应的正确拼写，建造一个无歧义的误拼字典。在进行英文单词拼写检查时，查找误拼字典，如命中，则说明该单词拼写有误，该词的正确拼写字段为纠错建议。该方法的特点是侦错和纠错一体化，效率高。但英文拼写错误具有随机性，很难保证误拼字典的无歧义性和全面性，因此查准率低、校对效果差。 |
| 词形距离法 | 这是一种基于最大相似度和最小串间距离的英文校对法。其核心思想是构造单词的似然性函数，如该单词在词典中，则单词拼写正确;否则，按照似然性函数，在词典中找到一个与误拼单词最相似的词作为纠错候选词。该方法的特点是节省存储空间，能反映一定的常见拼写错误统计规律，是一种模糊校对法。 |
| 最小编辑距离法 | 通过计算误拼字符串与词典中某个词间的最小编辑距离来确定纠错候选词。所谓最小编辑距离是指将一个词串转换为另一个词串所需的最少的编辑操作次数(编辑操作是指插入、删除、易位和替换等) 。还有人提出了反向最小编辑距离法，这种方法首先对每个可能的单个错误进行交换排列，生成一个候选集，然后，通过查词典看哪些是有效的单词，并将这些有效的单词作为误拼串的纠错建议。 |
| 相似键法 | 相似键技术是将每个字符串与一个键相对应，使那些拼写相似的字符串具有相同或相似的键，当计算出某个误拼字符串的键值之后，它将给出一个指针，指向所有与该误拼字符串相似的单词，并将它们作为给误拼字符串的纠错建议。 |
| 骨架键法 | 通过构建骨架键词典，在英文单词出现错误时，先抽取出该错误单词的骨架键，然后再去查骨架键词典，将词典中与该单词具有相同骨架键的正确单词作为该单词的纠错建议。 |
| N-gram法 | 基于n元文法，通过对大规模英文文本的统计得到单词与单词间的转移概率矩阵。当检测到某英文单词不在词典中时，查转移概率矩阵，取转移概率大于某给定阈值的单词为纠错建议。 |
| 基于规则的技术 | 利用规则的形式将通常的拼写错误模式进行表示，这些规则可用来将拼写错误变换为有效的单词。对于一个误拼字符串，应用所有合适的规则从词典中找到一些与之对应的单词作为结果，并对每个结果根据事先赋予生成它的规则的概率估计计算一个数值，根据这个数值对所有候选结果排序。 |

我们主要讲解最小编辑距离法。

## 最小编辑距离法

通过计算误拼字符串与词典中某个词间的最小编辑距离来确定纠错候选词。所谓最小编辑距离是指将一个词串转换为另一个词串所需的最少的编辑操作次数(编辑操作是指插入、删除、易位和替换等) 。还有人提出了反向最小编辑距离法，这种方法首先对每个可能的单个错误进行交换排列，生成一个候选集，然后，通过查词典看哪些是有效的单词，并将这些有效的单词作为误拼串的纠错建议。

1. 最小编辑距离法

## 编辑距离

编辑距离是指两个字串之间，由一个转成另一个所需的少编辑操作次数一个所需的少编辑操作次数。编辑距离应用于DNA分析、拼写检查、语音识别、抄袭侦测等领域。

许可的编辑操作包括：

将一个字符替换成另一个字符substitution；

插入一个字符insert ¾插入一个字符insert；

删除一个字符delete;

例如：kitten转成sitting：

1. sitten （k替换成s）

2. sittin （e替换成i）

3. sitting （最后添加g）

但是，注意到，一个单词转化为另一个单词所需的编辑操作次数是可变的，如：将through转成though：

|  |  |
| --- | --- |
| 方法一  thoough （r替换成o）  thouugh （o替换成u）  thouggh （u替换成g）  thoughh （g替换成h）  thouug （删除最后的h） | 方法二  though （删除r） |

可见两种方法的编辑操作次数相差甚远，因此，如果把所有的可能的编辑操作次数列举出来的话，穷举的空间是巨大的，所以，要求出最小的编辑距离，引进Levenshtein编辑距离。

## Levenshtein编辑距离

Levenshtein编辑距离是由俄国科学家Vladimir Levenshtein在1965年提出的，采用了动态规划的方法，目的是求出两个词之间的最小编辑距离。

算法描述：

Proc: LevenshteinDistance

Input: string a, string b

i = a.length()+1;

j = b.length()+1;

initialize matrix

m=0;

while m<i

do

matrix[m++][0]=m;

end while

m=0;

while m<j;

do

matrix[0][m++]=m;

end while

fill in the matrix

m=1; n=1;

while (m++<i)

do

while(n++<j)

do

dmn= matrix[m-1][n-1];

if(a.at(m-1)!=b.at(n-1))

dmn+=2;

end if

matrix[m][n]=minimum(matrix[m-1][n]+1, matrix[m][n-1]+1, dmn);

end while

end while

return matrix[i-1][j-1];

end proc

## 本实验中Levenshtein距离的应用

考虑到计算Levenshtein距离需要两个输入参数：一个错词，一个真词。当我们检测到一个错词的时候，如果把字典里所有的词都与该错词进行Levenshtein距离计算的话，显然不适合，所以我们必须从字典里取出一个真词子集，再计算他们的Levenshtein距离。

真词子集的提取：

首先想到的肯定是通过计算错词的词长，在取出字典里所有与其词长一致的单词。我们觉得，这是一个正确的做法。但是，还不够，因为日常生活中，我们输入错词的原因不仅仅是因为拼写错误，很多情况下，都是因为少输入或者多输入一个或多个字母而导致的，所以我们在提取子集的时候，必须设置一个表示词长相差多少的阈值，在一定范围内取出所有适当的真词，进行Levenshtein距离计算，再通过Levenshtein大小进行排序，最终返回给用户最优结果。

算法描述：

//设置5个标志，最多给用户返回5个候选词

int max=4, midmax=3, mid=2, midmin=1, min=0;

struct dis

{

int distance;

QString word;

}minidis[5]={{96,""}, {97,""}, {98,""}, {99,""}, {100,""}};

for all words in dictionary

do

//N 为设置的阈值

if(wrong\_word.length < this\_word.length + N ||

wrong\_word.length > this\_word.length - N)

subset.push\_back(this\_word);

end for

for all words in subset

do

curdis = LevenshteinDistance(wrong\_word, this\_word);

//排序

if(curdis<minidis[min].distance)

{

minidis[max].distance=curdis;

minidis[max].word=candidate.at(i);

//交换标志，因为效率比交换整个结构体高。

max=min; max=midmax; midmax=mid; mid=midmin; midmin=max;

}else if(curdis<minidis[midmin].distance)

{

minidis[max].distance=curdis;

minidis[max].word=candidate.at(i);

//交换标志，因为效率比交换整个结构体高。

max=midmax; midmax=mid; mid=midmin; midmin=max;

}else if(curdis<minidis[mid].distance)

{

minidis[max].distance=curdis;

minidis[max].word=candidate.at(i);

//交换标志，因为效率比交换整个结构体高。

max=midmax; midmax=mid; mid=max;

}else if(curdis<minidis[midmax].distance)

{

minidis[max].distance=curdis;

minidis[max].word=candidate.at(i);

//交换标志，因为效率比交换整个结构体高。

max=midmax; midmax=max;

}else if(curdis<minidis[max].distance)

{

minidis[max].distance=curdis;

minidis[max].word=candidate.at(i);

}

end for

//按Levenshtein距离从小到大返回候选词

return minidis[min, midmid, mid, midmax, max].word;

## 实际例题详细分析过程

单词：important

错误输入：inportant

1、计算得到的真词子集：

("ministry", "prompt day", "store-room", "thievish", "threnody", "certainly", "baffling", "summarily", "underpin", "Occidental", "laudable", "get into", "stranded", "materially", "prescient",......,"dependent", "magnesia", "accessory", "magnesium", "overstore", "consummate", "scrivener", "exogenous")

共10982个；

2、对该子集中所有单词，求该词与错误输入的错词之间的Levenshtein距离，部分结果如下：

"ungracious" 13

"assurance" 12

"sanctify" 11

"bath-house" 15

"undeniable" 15

"adjective" 14

...

"put right" 12

"important" 2

"hand-money" 13

"chancellor" 13

...

"imperative" 9

"enduring" 11

"informant" 4

"nutrition" 10

"abrasion" 11

"for ages" 11

...

取其中一个单词，分析其Levenshtein距离的计算过程：

真词：informant

a、初始化二维数组：

for(i=0;i<wrong\_word.length;i++)

M[i][0]=i;

for(j=0;i<real\_word.length;j++)

M[0][j]=j;

用一个表表示，看起来就像下面这样；

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9(t) |  |  |  |  |  |  |  |  |  |
| 8(n) |  |  |  |  |  |  |  |  |  |
| 7(a) |  |  |  |  |  |  |  |  |  |
| 6(t) |  |  |  |  |  |  |  |  |  |
| 5(r) |  |  |  |  |  |  |  |  |  |
| 4(o) |  |  |  |  |  |  |  |  |  |
| 3(p) |  |  |  |  |  |  |  |  |  |
| 2(n) |  |  |  |  |  |  |  |  |  |
| 1(i) |  |  |  |  |  |  |  |  |  |
| 0 | 1(i) | 2(n) | 3(f) | 4(o) | 5(r) | 6(m) | 7(a) | 8(n) | 9(t) |

表1

b、自下而上，从左往右，计算：

TMP=wrong\_word.at(i)==real\_word.at(j)? M[i-1][j-1]:M[i-1][j-1]+2;

D(i, j)=min(M[i-1][j], M[i][j-1], TMP);

计算结束后，二维数组看起来就像是这样子的：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9(t) | 8 | 7 | 8 | 7 | 6 | 7 | 6 | 5 | 4 |
| 8(n) | 7 | 6 | 7 | 6 | 5 | 6 | 5 | 4 | 5 |
| 7(a) | 6 | 5 | 6 | 5 | 4 | 5 | 4 | 5 | 6 |
| 6(t) | 5 | 4 | 5 | 4 | 3 | 4 | 5 | 6 | 5 |
| 5(r) | 4 | 3 | 4 | 3 | 2 | 3 | 4 | 5 | 6 |
| 4(o) | 3 | 2 | 3 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3(p) | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2(n) | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1(i) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0 | 1(i) | 2(n) | 3(f) | 4(o) | 5(r) | 6(m) | 7(a) | 8(n) | 9(t) |

表2

由此可得，inportant到informant的Levenshtein距离为M[9][9]=4。

c、对所有的结果进行排序，最多保留5个距离最小的结果，它们将会是：important、 ignorant、informant、in point、inconstant，最后返回给用户。

## 互联网查词功能实现

鉴于本次通查词功能过分依赖于本地词典，而本地词典大小有限，很难满足日常查词需要，故增加网络查词功能。通过微软必应词典给我们提供的强大的平台，我们可以轻松的解决大部分单词的翻译和例句查看的功能。

该功能并未利用Bing.cn给我们提供的简单的API，而是通过直接解析网页来完成。URL格式如下：

http://cn.bing.com/dict/search?q=word&go=&qs=n&form=CM&pq=word&sc=7-5&sp=-1&sk=

我们只需把q=word字段替换成我们需要查询的单词，如q=precious，在发送http请求，获取html文件，在通过正则表达式匹配，即可获取我们需要的信息。用到的正则表达式如下：

1. 用于提取解释的正则表达式：

QRegExp keyrx("<ul><li><span class=(\"pos\"|\"pos\\sweb\")>.\*</li></ul>");

1. 用于提取中文例句的正则表达式：

QRegExp cnrx("<div class=(\"bil\_ex\"|\"sen\_cn\")>.\*</div>");

1. 用于提取英文例句的正则表达式：

QRegExp enrx("<div class=(\"val\_ex\"|\"sen\_en\")>.\*</div>");

1. 用于去除<a>标签的正则表达式：

QRegExp noarx("(<a.\*>|</a>)");

一个简单例子（查询单词：precious）：

1. Http请求：

QNetworkRequest(QUrl("http://cn.bing.com/dict/search?q="+word+"&go=&qs=n&form=CM&pq="+word+"&sc=7-5&sp=-1&sk=")))

1. 提取到的html如下：

<span class="pos">adj.</span><span class="def"><span> 珍奇的；珍稀的；宝贵的；珍贵的</span></span></li><li><span class="pos">n.</span><span class="def"><span>可爱的人；【女名】女子名</span></span></li><li><span class="pos">adv.</span><span class="def"><span>强调极少或太少</span></span></li><li><span class="pos web">网络</span><span class="def"><span>珍爱人生；贵重的；宝贝</span></span>"

"<div class="val\_ex">a precious vase</div>"

"<div class="bil\_ex">稀世花瓶</div>"

1. 显示：

Qt QTextEdit控件给我们提供了强大的html解析功能，让我们可以直接用它来显示提取到的部分html文件。

1. 系统结果展示及分析

测试例文：

There is not dout that happiness is the most presious thing in the world. Without it, lift will be empty and meaningfulless. If you wish to know how to get happiness, you must pay attention to the following two points. Firstly, health is the sekret of happiness. Only a strong man can enjoy the pleashure of life.

系统界面：

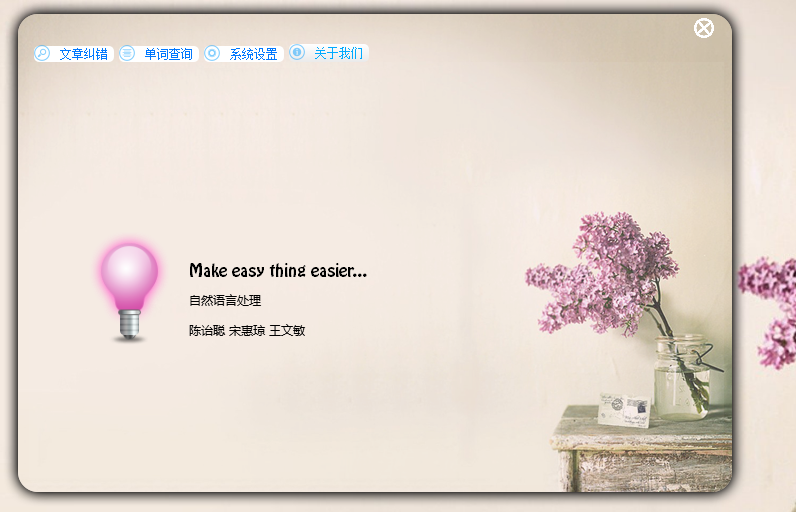


图 1

在文章纠错界面输入一段文章然后点击右下角的“检错”按钮，就会把拼写出错的单词标红：

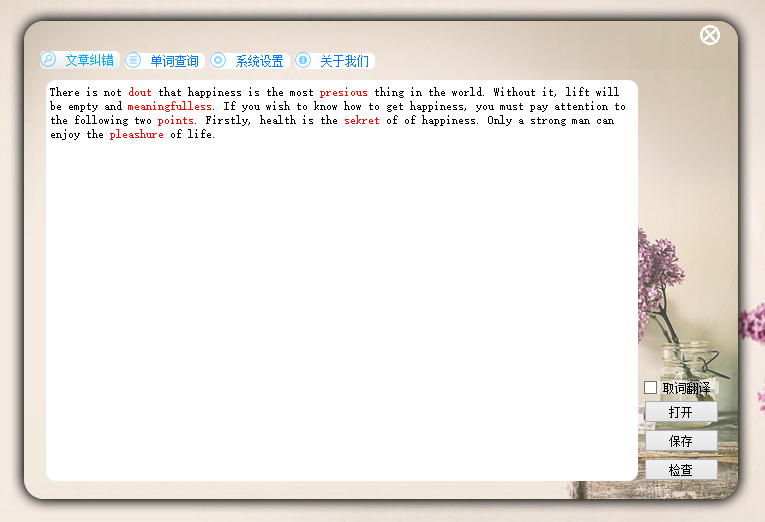


图 2

点击出错的单词，就会在旁边出现可供选择替换的单词：

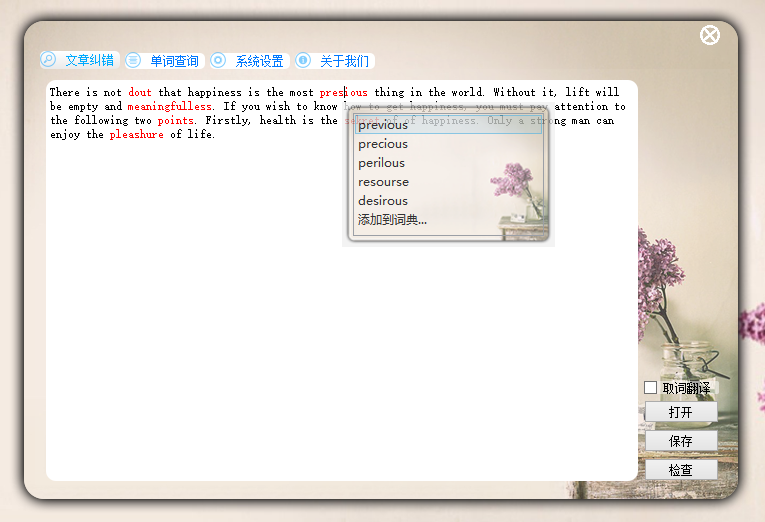


图 3

双击选择正确单词则替换后的单词会变成绿色的：

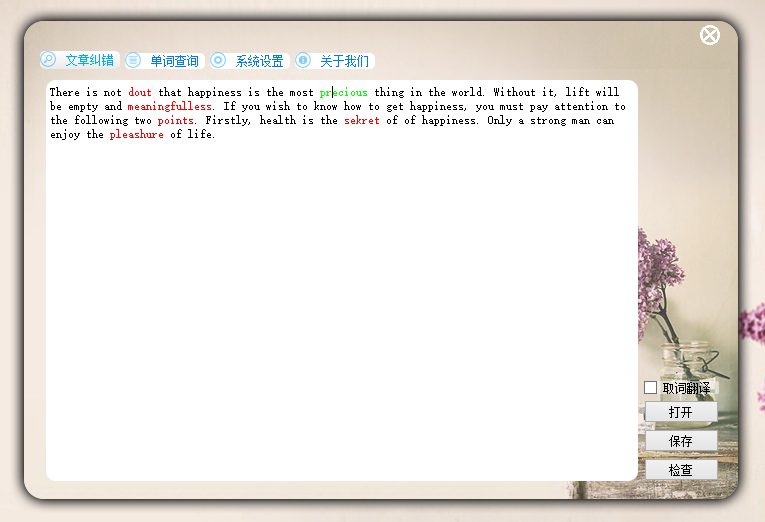


图 4

当我们不仅仅关心是否有错词的时候，我们可以把取词翻译选项勾上，然后点击编辑框内的任何单词，即可查看该词翻译：

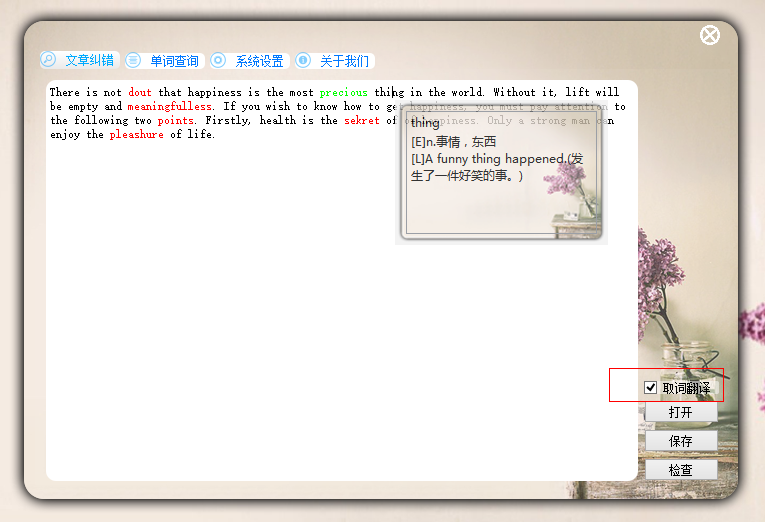


图 5

在单词查询界面输入单词即可查询该单词的中文意思以及例句：



图 6

如果你输入的单词不存在的话，系统将给出拼写建议：



图 7

当然，你也可以通过强大的互联网来查询：

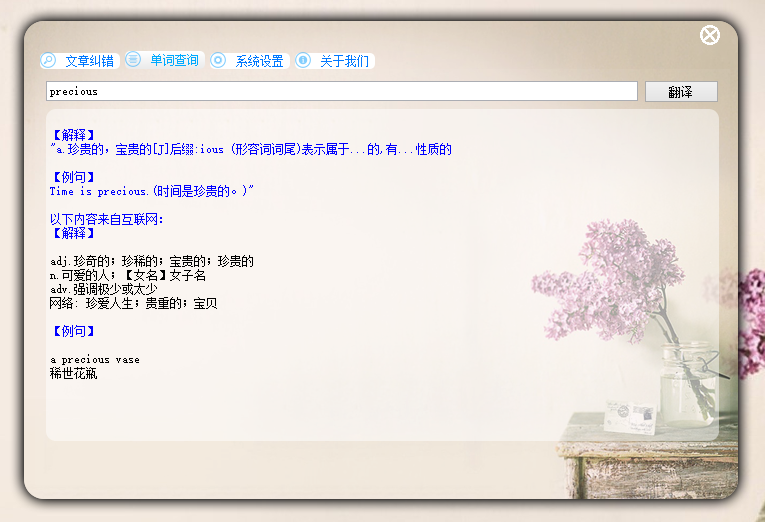


图 8

在系统设置界面可以手动添加或者批量添加新增加的单词，其中文翻译以及英文例句：



图 9

当然，你也可以通过文件进行批量添加，不过，系统对文件的格式有要求，必须满足如下格式：

NewWord[E]Explanation[L]SimpleSentence

新增加的单词可以在单词查询界面查询到相关该单词的用法：



图 10

1. 学习体会及对该课程的意见和建议

通过一学期对自然语言处理的学习，我们学习了解了分词、词法分析、句法分析、语法分析、拼写检错以及机器翻译等知识，也掌握了基于规则的分析方法和基于统计的分析方法、隐马尔科夫模型、最小编辑距离等理论知识，通过完成各个小作业以及大作业，既深入了解理论知识，又加强了编程能力，还学会与同学分工合作以完成一个更优秀的系统，让自己在大数据时代以及机器理解领域具备更多的知识以跟上互联网时代发展的步伐。

该课程对重要理论知识讲解透彻深入，让同学们比较好接受学习，而且设置了编程实验课，让我们把理论知识付诸实践实现小系统的设计和应用，既加强了解理论知识，又加强了编程能力，还学会与同学分工合作，这都是课程设计成功的地方。要是以后可以增加多一些课外自然语言知识的讲解或者讲解一些自然语言处理的相关新闻和新技术可以大大提高同学们的知识面，同时增加相关自然语言处理的实验课小比赛，会让同学们大大提升对这门课的热爱。

1. 小组成员打分

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 陈诒聪（组长） | 宋惠琼（组员） | 王文敏（组员） |
| 组内分工 | 1. 负责纠错系统的代码实现 2. 文档里面代码详解部分 | 1. 文档的编写与整合 2. 词典资料收集 | 1. 文档的编写 2. 词汇更新功能实现 |
| 评分 | 95分 | 90分 | 90分 |

1. 参考文献

《自然语言处理综论》Daniel Jurafsky / James H. Martin

《统计自然语言处理基础》Chris Manning / Hinrich Schütze

《统计学习基础》Hastie.T.

《自然语言理解》 江铭虎 高等教育出版社