拼写纠错

原理介绍

拼写纠错的原理就是对一个特定的词w，找出他最可能的正确形式。（这里存在一个问题就是如何判断这个词是错误的，为了简单起见，就将没有出现在词典中的词看作是错误的。）因此拼写纠错的任务就是从所有的候选集中找出概率最大的正确形式，记为c。使用数学的方式表示为：

$$\arg\max_{c\in\text{candidates}} \mathrm{P}(\mathrm{c}|\mathrm{w})$$

使用贝叶斯公式可以将上式转化为：

$$\arg\max_{c\in\text{candidates}} \mathrm{P}(\mathrm{c})\mathrm{P}(\mathrm{w}|\mathrm{c})/\mathrm{P}(\mathrm{w})$$

因为对于候选集中的元素来说，$\mathrm{P}(\mathrm{w})$是相同的，因此可以忽略，故最后要求的目标为：

$$\arg\max_{c\in\text{candidates}} \mathrm{P}(\mathrm{c})\mathrm{P}(\mathrm{w}|\mathrm{c})$$

这个表达式包含了四个部分：

- Selection Mechanism：argmax，即选择能使得该组合概率最大的candidate
- Candidate Model：$c \in candidates$，明确哪些是需要考虑的
- Language Model：P(c)。这里使用语言模型是为了使得纠正之后的词语c更符合语境
- Error Model：P(w|c)，表示正确的词语c被写作w的概率。如P(teh|the)的概率明显会大于P(theeexyz|the)的概率

对于这个表达式的直观理解就是对于一个写错的w，要综合考虑可能是要打哪个c和这个c是否符合当前上下文语境。

具体实现

接着是对这四个部分的具体实现。

Selection Mechanism

选择机制只需要使用python的内置函数argmax即可

Candidate Model

Candidate Model是指错误的w真是想要表达的c，因此c是又w通过

- deletion：删除一个字符
- transposition：交换两个相邻的字符
- replacement：替换一个字符
- insertion：在字符中间添加一个字符

得到的，通常使用**编辑距离(edit distance)**来衡量这几种操作。

在这里我们通过已知的w来通过若干次变换得到c。基本函数为：

```
def edits1(word):
    "All edits that are one edit away from `word`."
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(word[:i], word[i:])    for i in range(len(word) + 1)]
    deletes    = [L + R[1:]               for L, R in splits if R]
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]
    replaces   = [L + c + R[1:]           for L, R in splits if R for c in
letters]
    inserts    = [L + c + R               for L, R in splits for c in letters]
    return set(deletes + transposes + replaces + inserts)
```

得到的结果再调用一次edits1即可得到两次变化后的集合(也有可能变换为原始集合)。

这里需要注意的地方是最后返回的是一个**set而不是list**，set在某些情况下可以加速查找速度。list的查找是线性的，而set的查找在某些情况下是log(N)的。

这并没有完成我们所需要的编辑结果，这里需要对返回的结果进行过滤，因为在上文中我们假设错误的单词是词典中没有的单词，因此在这里我们将得到的集合中字典没有出现过的单词过滤出去，这样会大大减少原来的集合。

Language Model

对语言模型的建模即根据数数(最大似然估计+简答的平滑)的方法来构建一元、二元模型。

Error Model

对于Error Mode即要搜集用户拼写错误的语料，通过最大似然估计(计数)的方法来计算P(w|c)。

进一步的改进

1. P(c)，即Language Model。对于Language Model来说无法避免未登录词的影响。因为一元模型建模的时候就是数一个词在字典中出现的个数。如果字典中没有这个词，那么我们就不可能将这个正确的词找出来。对于这个问题改进的方法有：
    - 搜集更多的语料
    - 使用一些英文语法，如在词尾增加"ility"或者"able"之类的形式
    - 对Candidate Model不进行过滤，允许出现没有见过的词语
2. 结合上下文语境，这个很好理解，如果单看一元模型（unigram）的话，只能选择出现频次较高的，但是结合上下文语境明显更符合常理，因此这也是对语言模型的一种要求。
3. 最后是对速度的提升，即实现“编译”而不是“解释”，就是缓存计算结果来避免重复计算。

参考

编辑距离算法实现

拼写纠错简单实现

paper1

paper2