人工智能导论
拼音输入法作业报告

齐强 2017011436 计 75

2019 年 4 月 21 日

目录

1 快速使用

1. note: 以下命令需要在 src 文件夹下运行
2. help: python3 main.py -h
3. train: python3 –train –data {指定训练数据 path}
4. valid: python3 –input {input file path} –output {output file path}

2 文件结构

1. src: 包含正式代码与自动测试相关代码
   - common.py: 包含通用的设置，包括模型相关文件 all-ch.txt, pinyin2ch.txt 和模型的路径、总字数的数量等
   - model.py: 包含 Model 类 (代码主要部分)，根据输入进行训练或预测
   - main.py: 解析选项，决定训练或预测，解析 inputfile，将一行的拼音转成 list 后调用 model predict() 预测，并将答案输出到 outputfile
   - train.sh: 训练脚本（新闻文件有点多，手动训练有点慢...)
   - autotest.sh | gen_text.py | gen_py.py | valid.py 自动测试相关
2. data: 包含存储输入输入文件
3. model: 包含模型相关文件
   - all-ch.txt: 给定文件，确定所有进行统计的汉字
   - pinyin2ch.txt: 给定文件，根据拼音确定相应可能汉字
   - dict.json: 用于存储训练时与测试时需要用到的常用 dict (省去部分准备时间)
   - model.npz: 模型文件，采用 numpy 的数据存储格式，包含 1-gram, 2-gram, 3-gram 组的出现频数

3 算法思路

1. 最终实现了字的二元、三元相结合的算法, 同时保留了单二元的方法
2. train part:
   - 遍历给定数据，统计所有单字出现的频数与所有 2 元组出现的频数，将结果分别保存在 c1, c2 数组中
   - 取出 c2 数组中记录下的前 6763 高频的 2 元组，重新遍历训练数据，统计以这些高频 2 元组 + 其他字做 3 元组的出现频数
3. predict part:
   - predict(): 2 元预测
     $prob_{i,j} = max(prob_{i-1,k} + log(N(\omega_{i,j}\omega_{i-1,k})) - log(N(\omega_{i-1,k})))$

- predict2(): 2 元 + 3 元预测
  $if\ \omega_{lk}\ in\ [highfrequency2-gramword]:$
  $prob_{i,kj} = max(prob_{i-1,lk} + log(N(\omega_{i-2,l}\omega_{i-1,k}\omega_{i,j})) - log(N(\omega_{i-2,l}\omega_{i-1,k})))$
  $else$
  $prob_{i,kj} = max(prob_{i-1,lk} + log(\omega_{i-1,k}\omega_{i,j})) - log(N(\omega_{i-1,k})))$

4 测试效果

写了一些自动测试代码，从新闻训练数据中找出一定数量文本，并由句号、逗号分割产生粗糙训练文本，由粗糙训练文本产生拼音数据文件与正确答案文件，根据 main.py 产生的预测结果与 gen_py.py 产生的正确答案进行对比，得出模型准确率

seq	total line	accuracy line	total word	accuracy word
0	131	0.2977	1885	0.8233
1	153	0.1699	2266	0.7691
2	196	0.2102	3059	0.8394
3	167	0.2455	2135	0.8018
4	100	0.2300	1436	0.7994
5	212	0.2783	2304	0.8085

5 总结收获

1. 问题：
   - 无法对多音字进行判断，导致训练时会存在偏差
   - 因为时间有些来不及，没有更好的调参，对于平滑处理等不是很好
2. 总结：
   - 2 元与 3 元结合能够提升一些名词较长的句子的准确率
   - 训练语料库与测试集的相关性，对于准确率有着非常大的影响，增加训练语料库会提高一定的正确率