中文纠错比赛系统报告

语法小能手团队：刘小辉[1]，黄小钒[1]
[1] 北航，软件学院

Abstract

本文针对*AI studio*上中文智能纠错比赛，总结本团队的具体实现方案以及分析和调优过程。总结即包括一些成功的实践,也包括一些失败的尝试经验。我们将中文纠错问题划分为两类问题：拼写问题和非拼写问题。将训练集中*source*句子与*target*句子等长样本划分为第一类拼写问题样本，具体包括音近、形近错误，乱序错误等，将不等长的样本作为非拼写问题，具体包括：缺字问题和冗余字问题，方便了模型集成效果的提升。我们实现的项目主要组成结构为：文档资料模块、数据扩增模块、词库等知识库模块、模型模块。其中我们的模型模块包括了：*pycorrector*的传统纠错算法的改进，比赛方*baseline*模型*MiduCTC*，*pycorrector*提供的*macbert*模型，*ECSpell*模型。我们探索并实现了对这些模块的最佳集成方案，也尝试对这些模型进行了修改，其中对*pycorrector*传统算法的修改版有效提升了模型集成效果，也尝试了对*MiduCTC*模型表征方面尝试融合词性以及拼音，并添加到*vocab*词库中，重新训练发现模型预测能力大大降低。词库知识库模块包括：网上收集的*200*万词库以及对应的拼音-词映射库，成语诗词专名词库，腾讯词向量，混淆集，易错词等，*200*万词库所对应的拼音-词映射库搭配成语诗词专名词库用于专名词错词检索召回。数据扩增模块我们尝试了继续对比赛训练*target*正样本进行音近和形近错词错字替换，但训练效果并不理想。本文将分别从以下几个方面对这些模型集成及优化思路进行说明：

- 利用*jieba*分词去除重复字词
- *pycorrector*传统纠错算法改进
- 模型训练
- 模型预测召回
- 模型集成
- 误纠检测与还原

1. 模型构建

模型总体流程图如图1所示。系统目录结构如图2所示。

利用jieba分词去除重复字词. 通过分析数据集样本，发现存在大量相邻的重复字词问题，针对此类问题，我们使用了jieba分词，然后将分词后前后相邻的两个词，若至少存在一个词的字数大于1，前另一个词是这个词的相邻端的子串，则判断存在冗余重叠字词，并去除较短的那个重叠词。经过测试验证，对比了不同分词器，使用jieba分词+自定义200万词库作为分词词典，有较明显效果提升。

pycorrector传统纠错算法改进. pycorrector作为开源中文纠错项目框架，集成了传统规则算法和基于统计模型的算法，包括了混淆集和专名词的替换功能，另外集成了几种深度学习中英纠错模型，最开始，我们直接使用pycorrector传统纠错算法，经过测试，原pycorrector传统算法纠错得分很低，f1得分在12%左右，为了利用传统规则算法及混淆集，专门词检测替换功能，在使用专名词替换逻辑方面，我们通过分析代码实现逻辑，发现原来n-gran循环所有n-gran词，在专名词较长时计算速度慢，另外召回的词多而不准的问题，修改为首先对文本进行分词，将分词后的词当做一个整体不在拆分，大大降低了计算复杂度，另外为了避免分词的不准确导致的漏召回，我们又将jieba分词替换为LTP分词模型，对比发现由于LTP分词粒度更细更准确，减少了被误过滤的组合词，提升

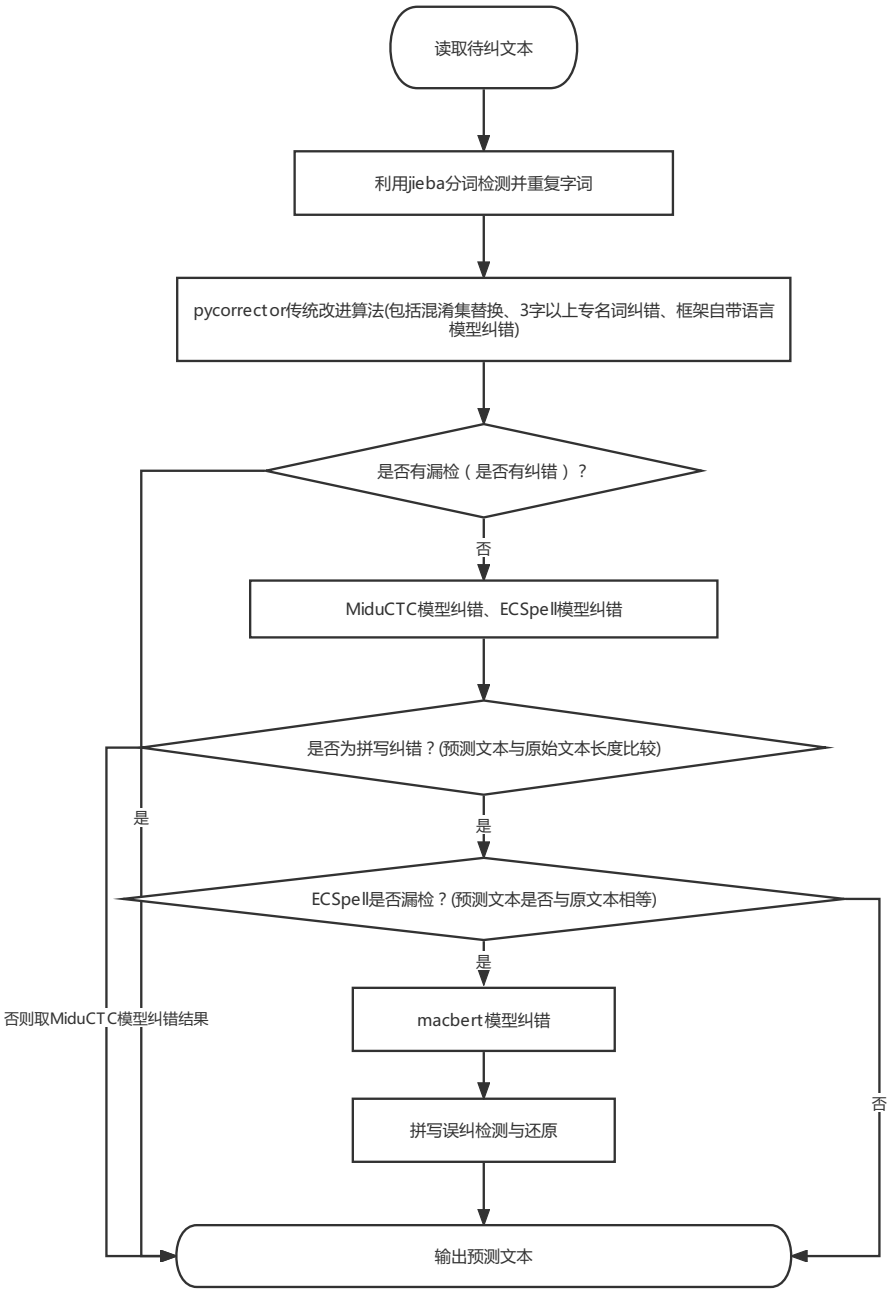

图 1. 中文纠错总体流程

了构造匹配词召回率和准确率，然后针对分词的词组，我们构造了4-8字范围长度的待匹配gran词，在搜索匹配专名词库时，匹配顺序上优先从长到短进行匹配，当较长的词匹配到专名词，其相似度达到阈值则召回，且后续子串不在参与匹配。匹配方式上，为了召回专名词单字拼写错误、缺字错误，以及相邻字乱序错误，我们构造了词拼音到词的映射字典库，我们为了能够实现同音词，但容忍词中任意一个字的音不同，以及

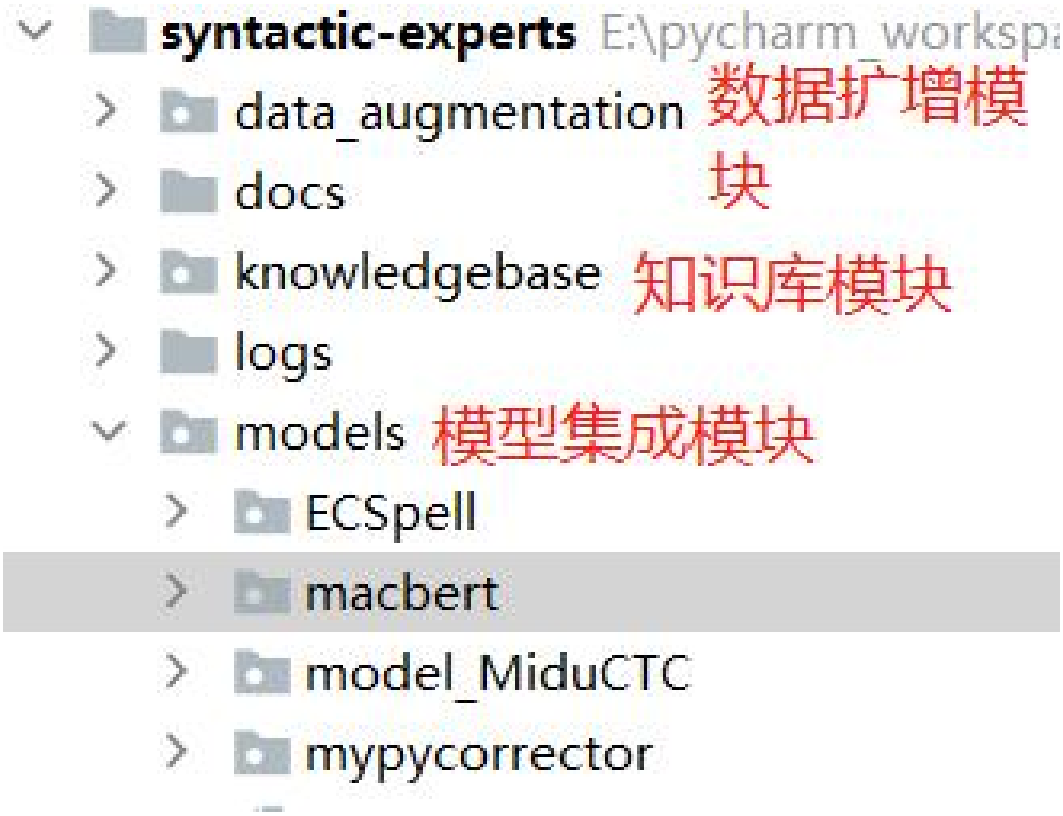

图 2. 系统目录结构

容忍一个字的缺失，依次构造了多组覆盖了其中一个拼音字的拼音-词映射键值对，以空间换时间的方式加快了同音或近音词中存在不同字、缺字或相邻两字乱序的专名词搜索召回。为了降低误召回的专名词数量，我们又进行了以下修改：首先，对匹配出的召回集排序，考虑到原有的匹配词与召回词若存在不同字或缺一个字时，算出的得分为0而被过滤的情况，我们对每一个匹配词所对应的召回候选专名词集合提出了一种排序规则，以选出最佳候选专名词，即是对每个召回的专名词算出一个综合得分：

$$score = score_{sim} + score_{shape} + score_{replace} \quad (1)$$

其中$score_{sim}$为原音近形近得分，$score_{shape}$为形近得分，$score_{replace}$使用腾讯词向量，分别计算替换词与原词前后词的相似度，以及原词与前后词的相似度的差值。只对该匹配词对应的召回集中top1进行计算综合得分，不管是否符合阈值，该召回集中的后续词不在参与计算得分。其次，为了进一步降低同音但容忍一个字不同音以及一个字缺失的专名词搜索误召回率，排除200万词库中低专名词的匹配，我们对整个训练集的target样本运行了专名词匹配算法，并将预测文本与原target词对比，当替换的专名词实际不对时，将误替换的原始词及对应专名词以键值对的形式保存到一个低专名词文件中，该低区分度的的专名词库中的词不在参与不同音词的匹配（即只进行全部同音词匹配，并计算和判断最后的得分阈值）。最后，如果原始匹配词本身是专名词则不在参与专名词匹配。经过次算法，我们保证了被召回的结果集中，正确召回的样本数明显大于无召回的样本数，测试验证比例在5:1左右。

模型集成. 我们对比了每个模型的预测效果，发现召回率较低，但预测精确率高，故模型集成时考虑了模型互相漏检时取存在纠错的模型预测结果，另外，MiduCTC模型针对非拼写问题纠错能力明显比其他模型能力更强，而拼写问题纠错，则ECSpell模型更好，故将MiduCTC预测为非拼写问题纠错时使用MiduCTC预测结果，否则取ECSpell模型结果，当ECSpell的漏检，且macbert召回模型存在纠错时，才取macbert召回模型的预测结果。其中macbert召回模

型是对macbert预测结果top20中通过计算了音近形近以及腾讯词向量相似度得分来排序得到最终预测结果，不过由于本身top20中正确率就相对较低，故相比macbert预测top1的结果，收效甚微。

误纠检测与还原. 为了减少深度模型集成时，误纠率，我们增加了一种误纠检测还原方法，即简单的通过LTP分词原句与模型集成预测句子的词组长度，当纠错后的句子分词的词组数组长度更长则还原输出为原始句子。经过测试，并对比了不同的分词方式，使用LTP分词有一定提升效果。

2. 实验结果

模型训练. 模型的训练分为两个阶段，第一个阶段是对原始训练集的训练，第二个阶段是对验证集的微调训练。训练集大概训了4个epoch，验证集上微调则在1-6内不等，学习率基本是在5e-5，微调验证集时，我们为了防止过拟合，除了控制epoch的个数，还采用了多个验证集相互训练的阶段，每100个step产生一个checkpoint，然后依次对每个验证集上进行评测，我们选择的checkpoint模型文件根据是否在每个验证集词和句子的f1得分是否都比较均衡的原则。比如ECSpell模型的微调选择checkpoint过程，根据评测结果，我们选择checkpoint-300，评测checkpoint如表1。

F1 checkpoint / 训练集	200	300	400
final_val	0.57,0.58	0.63,0.64	0.69,0.69
final_train	0.66,0.68	0.65,0.67	0.63,0.65
preliminary_extend_train	0.65,0.66	0.65,0.65	0.62,0.63
preliminary_val	0.73,0.73	0.73,0.73	0.72,0.73

表 1. 模型checkpoint在验证集上的微调评测

3. 总结与展望

此次比赛我们尝试了各种模型集成方式，分析了验证集的错误类型，最终的集成方案分别在A榜，B榜及决赛分别获得0.527,0.5317,0.56的成绩，我们也有过一些失败的尝试，包括了对MiduCTC模型将词性和拼音引入表征，但训练后得分降低的情况，原因可能是引入的新特征破坏了原有预训练模型的参数特征，原有模型的表征改变不适合做微调。另外我们尝试了macbert的预测结果topK召回，但召回率较低，最后，我们发现我们的模型集成方面，针对缺字问题以及乱序等非拼写问题预测得分比较低，故尝试对MiduCTC模型进行召回，发现召回正确数较高，但由于时间关系还没有找到从这个召回结果集中选择正确结果的办法。未来我们将尝试增强MiduCTC模型对非拼写问题的纠错能力，另外尝试训练一个错误类型分类模型以更好的对不同类型的错误进行处理和集成。

4. 参考地址

本团队实现的纠错系统所参考的github项目代码地址整理如下：

https://github.com/bitallin/MiduCTC-competition.git

https://github.com/shibing624/pycorrector.git

https://github.com/Aopolin-Lv/ECSpell.git