汉语文字自动检查系统的实现与分析

林海涛 201828014628036

卢宇 201828014628033

张肖寒 201828014628058

一、任务介绍

我们组的大作业选择的课题是：设计实现一种汉语文字自动检查系统（Chinese text proofreading），这类任务也称 Chinese Spelling Check（后称 CSC）。具体地，给定任意未分词的文本，通过算法找到错误的字符并进行改正。如图 1 所示，一般的错误包括同音不同字（“第”→“弟”）、字形相似（“奴”→“女”）、读音相似（“应”→“因”）等等，每个文本中错误个数是不确定的，也有可能没有错误。

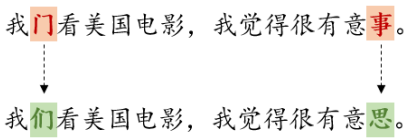

图 1. CSC 错误实例

每年都有这类主题的论文发表，但数量很少，使用的方法也有限。目前主流的方法是将整个算法看成一个 pipeline 的工作：找出错误的位置、计算出正确的替换字符。前者依据“错别字会带来异常的分词结果”，通过分词结果来搜索错误位置；后者主要是在开源的 confusion set 内寻找合适的替换词，主要方法是计算替换词的语言模型分数。我们的模型在实现了已有算法【1】的基础上，根据实验结果分析已有算法的不足，提出了新的错误位置检索方法，我们的方法在准确率和检索效率上都有很好的效果。

目前已有的通用数据集是 SIGHAN 发布的数据集 sighan7 和 sighan8，部分论文沿用 CLP2014 的数据集。算法的评估一般包括两个部分：是否可以找到错误的位置、是否可以正确修正，每个部分再由多个子指标评价，在第四章实验部分将进行详细介绍。

二、相关工作

在往届 SIGHAN 上，十余支队伍对 CSC 任务进行测评，绝大部分方法都是将任务拆分成两个子任务：错误位置识别、检索正确替代字符。

对于错误位置的识别，一旦某个文本出现了错别字，那么其前后分词结果就会发生变化。例如图 2 中，因为出现了错别字“门”，“我们”会被划分为“我/门”。方法【2】考虑了当前字符的部首、Unicode 编码、字符是否是数字、是否是标点等多个特征进行 CRF 分词。如果分词之后出现多个连续的单个字符，如“我/门/看”，则被看作是疑似错误位置。方法[3]直接分词，如果句子中含有超过 2 个连续的字符，则当作备选错误位置。方法[2]首先用模板进行识别匹配一些典型的错误，然后进行分词，然后将判断错误位置转化成一个序列标注问题，标签分别是“c/正确 w/错误”。

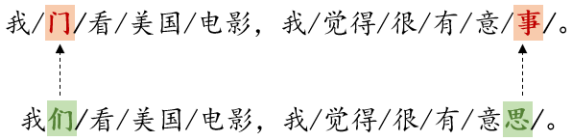

图 2. CSC 错误分词结果

对于正确字符的替换，绝大多数都是根据 confusion set，计算不同替换词之后的语言模型分数，从而选出得分最高的替换词。confusion set 囊括了各种类型的错别字，但由于词典较大，平均每个字都十余个候选项，所以检索的正确和高效十分重要。方法【2】提出了动态规划的算法，大大提升了运算效率。另外针对不同粒度的词语，采用不同的 n-gram 语言模型，方法【3】对所有词进行扫描，如果所有词长等于 2，使用二元语言模型，否则使用三元语言模型。

另外还有很多相关的工作，方法【4】提出加入 SMT 直接翻译错误句子，最后加入 SVM 从而 reranking 备选答案；方法【5】提出 OCR 和 ASR 的办法，将 OCR 识别错误的字符对作为训练数据；方法【6】提出基于规则的识别方法。

本次实验主要是基于方法【1】的思想，将错误位置的识别转化成标注问题。但是在实验中，我们放弃了论文中提到的 CRF 的方法，而换成了 SVM，后续我们将详细论述原因并分析我们方法的合理性。

三、模型设计

如图 3 所示，整个模型分为以下几个步骤：分词、运用 SVM 寻找错误位置或者直接根据分词结果判断、根据 confusion set 找到备选答案、根据语言模型找到得分最高的答案。后续将主要从“错误位置识别”和“检索正确替代字符”两个角度来介绍。

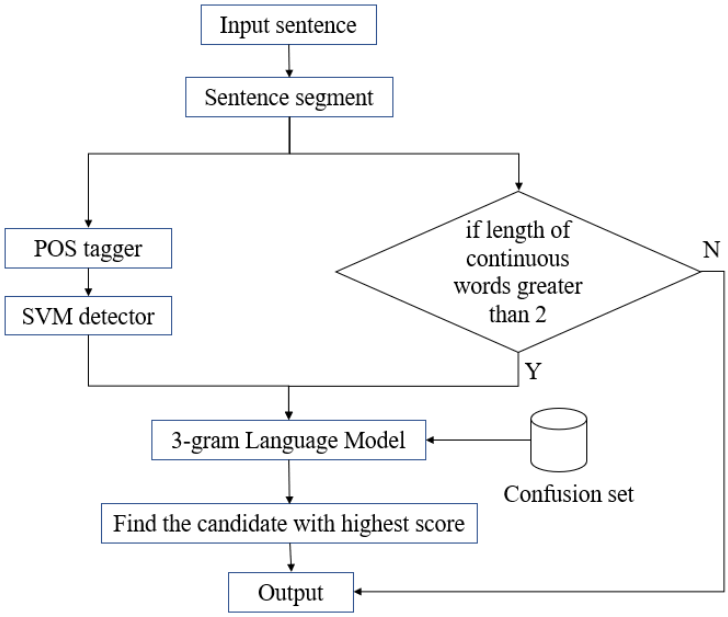

图 3. 模型流程图

1、 识别错误位置

对于一个给定的句子，我们第一步的目的就是找出错误的字可能的位置。针对于这一个问题，我们设计了两种不同的方法来进行判断。(1)基于分词的无监督的方法 (2)基于 CRF 的有监督的方法（3）基于 SVM 的有监督的方法。下面我们就对这两种方法分别作介绍。

（1）直接分词

一般来说，一个句子中错误的字大概可以分为两种，一是由于字音字形相近导致的错误；另一种是词语搭配用法错误。本次实验主要考虑的是如何找出并改正第一种错误，而字音字形导致的错误往往会让该字在分词结果中和上下文隔开。例如：“在/逆境/中/成长”和“在/逆/竟/中/成长”。因此我们考虑将分词过后被分成连续两个单个字的词加入候选的错误字列表当中。我们在 SIGHAN7 的测试集上进行测试，发现有 80%的错字是符合这种条件的。另外有些情况下，即使是错误的字也可能在分词的过程中和上下文分在一起。因此，和【3】类似，我们考虑使用一个词典来对这一类情况进行过滤，即分词结果中两个字或三个字的词不在给定词典中，那么我们也将其放入错误候选列表中。

另外，我们考虑出现连续错误的情况。如果考虑连续的错误，则会提高模型备选错误的

候选数量，但相应的改错的召回率也会提高。而不考虑连续的错误会提高模型计算的效率。两者各有利弊，因此我们在实验中对两种选项都进行了测试与对比。

（2）CRF

根据方法【1】的思想，首先使用上文提到的分词工具进行分词，然后将识别错误位置转化为一个序列标注的问题，选取了以下 3 个特征。

特征	含义
词语长度	“美国”的长度是 2
词性	共有 50 余种，如“形容词-a”、“副词-ad”、“方位词-f”等等
词义类别	对每个词取 300 维词向量进行聚类，得到 512 类

表 1. CRF 特征及含义

	precision	recall	f1-score	support
Correct	0.99	1.00	0.99	2000
Incorrect	0.00	0.00	0.00	24

表 2. CRF 测试结果

但经过实验，存在两个主要的问题：

1- 标签比例分布问题

现有数据集中每句话的错别字个数较少，平均每一句 1-2 处，有的句子也会出现没有错误，而通常一个句子词个数多达几十个，那么正负例的比例严重失衡。如表 1 所示，测试集正负例比例达到了 83：1，在用 CRF 处理的时候几乎没有办法识别负例。

原文中也提到了这个问题，并且采用了阈值的方法，即当被判定为正例的概率大于 0.99 时才会被判为正例。但我们实现时只能得到 CRF 的维特比解码过程中，在每个位置不同标签的最大分数，而得不到概率，如果简单地将分数进行归一化得到概率，那么最后选择标签时又会涉及不同节点之间关联的问题，所以在这一方面难以实现。

为了使标签更为平衡，我们对句子进行了剪裁，即以错误位置为中心，向两边剪裁子句，但是这样操作最后的效果还是不好。

2- 看作是序列标注问题是否合适

由问题 1 遇到的情况，我们怀疑 CSC 是否和序列标注问题很契合。首先，文本中错别字的位置是随机的，错别字的出现确实会带来异常的特征分布，但这种影响几乎是都是一阶的，而且也是很随机的。例如在图 2 中，“有/意思”被误写为“有/意/事”后，“事”并不会影响临近词“有”的特征。

（3）SVM

我们还是选择 CRF 中的特征，假设不同节点之间都是独立的，直接看作是分类问题进行训练。我们选择 sighan8 的 dev 集合作为训练集，并切分了验证集。为了实验周围节点的特征对错别字的判断是否有利，我们进行了如下对比实验。

实验	窗口大小			正负例比例		
	0	1	2	1：1	1：2	1：3
1	√			√		
2		√		√		
3			√	√		
4	√				√	
5	√					√

表 3. SVM 对比实验设置（window=0 即只取当前词的特征，window=1 则取当前词和前后各一个词的特征）

	Incorrect			correct		
	precision	recall	f1-score	precision	recall	f1-score
1	0.12	0.86	0.20	0.99	0.61	0.76
2	0.15	0.85	0.25	0.99	0.71	0.83
3	0.06	1.00	0.10	1.00	0.00	0.01
4	0.24	0.59	0.35	0.97	0.89	0.93

5	0.27	0.49	0.35	0.97	0.92	0.94

表 4. SVM 对比实验结果

从表 4 中可以看出，上下文的特征对于判断一个词是否有错别字是有用的，但是只有邻近的一个词是有效的，邻近的两个词就无法奏效。对于六个指标，我们更在意错别字的召回率，所以最后选择实验 1 的配置进行后续实验。

Dataset	Incorrect			correct			每句平均检测词个数
	precision	recall	f1-score	precision	recall	f1-score	
CLP2014	0.05	0.81	0.09	0.99	0.67	0.80	11.47
Sighan8	0.16	0.85	0.27	0.99	0.71	0.82	6.82
Sighan7	0.06	0.94	0.11	1.00	0.60	0.75	21.05

表 5. SVM 测试结果

由表 5 可见，SVM 在三个数据集上都可以保证很好的召回率。另外，由于 sighan7 数据中错别字数量更多，所以平均每句话检测的词个数也更多。

2、 错误字替换算法

由于 SIGHAN7 数据集中提供了易混淆的字的替换集，因此，针对每一个候选位置的字，我们把其所有可能替换的字都加入候选集合中，然后比较替换前和替换后的句子哪一个对应的语言模型得分更高，从而进行替换。

3、检索正确替代字符

（1）词向量模型

词向量方法背后的思想是，一个词的语义由它的上下文决定。Skip-gram 方法训练某个词的词向量时，将这个词记为 C，它的上下文词汇记作 O，训练目标就是最大化条件概率：

$$P(O=o \mid C=c)=\frac{\exp(u_o^T u_c)}{\sum_{w \in Vocab} \exp(u_w^T u_c)}$$

在此基础上，拼写正确的词应该会比拼写错误的词更有可能出现在原句的上下文中。比如，错误句子为“遇到逆竟时”，它的分词结果为“遇到/逆/竟/时”，将“竟”替换为“境”后，

分词结果为“遇到/逆境/时”，“逆境”在上下文中出现的概率，理论上要高于“竟”在上下文中出现的概率。

除了替换字符所在的词外，我们还考虑了其他词在上下文中的条件概率。因为在正确的句子中，每个词的条件概率都应大于在错误句子中的条件概率。因此，我们将每个词分别作为中心词计算概率，将句子中计算所得的所有词的概率相加，得分高者，就更有可能是正确的句子。

具体实现的时候，计算 softmax 的分母需要计算中心词和词表中所有词汇的内积，在词表比较大的时候需要耗费大量的时间和计算资源，难以实现。将条件概率取对数之后，公式可以表示为：

$$\log P(O=o \mid C=c) = u_o^T u_c - \log(\sum_{w \in Vocab} \exp(u_w^T u_c))$$

考虑到分词后，除错字外，句子中其他词大多数都是相同的，因此我们只取第一项。此外，由于分词后词的个数不同，因此，最终求和后要除以词的个数进行归一化。我们采用的词向量来自文献【7】中通过百度百科语料训练的词向量。

（2）N-gram 语言模型

我们选择 LDC 的中英双语语料的中文单句进行训练，为了考虑不同粒度的词语，我们训练了字符级别 bigram 和 trigram 的语言模型。最后的实验中我们选择 trigram 语言模型选择备选字符。语言模型得分越高，被选择的可能性越大。

因为代码运行速度还是可以接受的，我们直接对文本的 bigram 和 trigram 进行统计。

四、实验及结果

1、数据集：

本次实验中我们主要使用了 SIGHAN7 和 SIGHAN8 两个数据集。两个数据集的测试集分别有 1000 和 1100 个句子。SIGHAN8 数据集还提供了超过 1500 个例子作为训练集。两个数据集稍有不同，SIGHAN7 的数据每一个句子都是至少有 1 个错字，平均每句话错别字的数量为 1.26。而 SIGHAN8 的数据中有些句子是不含错别字的，平均每句话的错别字数量为 0.65。另外 SIGHAN7 包含两个子任务，subtask1 和 subtask2，分别是错误字位置检测和错误字改正。这里我们选用 subtask2，既可以考虑错误检测的效果，也可以考虑到错误改正的效

果。句子最初都是繁体中文字符，我们为了简化问题便于分析结果，将繁体中文全部转化为简体中文后再做分析。

2、评价指标：

根据 SIGHAN7 和 SIGHAN8 的比赛设定，我们使用检测准确率（DP），检测召回率（DR），检测 F1 值（DF），改错准确率（CP），改错召回率（CR），改错 F1 值（CF）六个指标来反映模型算法的效果。其中前三个反映的是算法发现错误位置的能力，而后三个反映的是算法改正错误的能力，结果越高代表模型的效果越好。

3、实验结果：

表 6、7 给出了不同的算法在两个数据集下的结果。single 代表考虑连续错误，consec 代表不考虑连续错误。svm 代表错误位置候选采用有监督的支持向量机的方法。word2vec 代表语言模型使用词向量相似度的计算方法，而 3-gram 代表语言模型使用 3 元组概率的计算方法。从实验结果来看，可以分析得出以下几点结论：

	DP	DR	DF	CP	CR	CF
Single+word2vec	0.110	**0.557**	0.183	0.052	0.262	0.086
Single+3-gram	0.300	0.509	0.378	0.235	0.400	0.296
Consec+word2vec	0.090	0.431	0.148	0.055	0.264	0.091
Consec+3-gram	**0.351**	0.470	**0.402**	**0.275**	0.368	**0.314**
Svm+word2vec	0.132	0.318	0.187	0.064	0.154	0.090
Svm+3-gram	0.215	0.534	0.307	0.164	**0.408**	0.234

表 6 不同方法在 sighan7 subtask2 上的实验结果

	DP	DR	DF	CP	CR	CF
Single+word2vec	0.119	0.411	0.184	0.040	0.138	0.062
Single+3-gram	0.260	0.317	0.286	0.180	0.220	0.198
Consec+word2vec	0.089	0.330	0.140	0.033	0.120	0.051
Consec+3-gram	**0.308**	0.284	0.295	**0.212**	0.196	0.204
Svm+word2vec	0.144	**0.446**	0.218	0.045	0.140	0.068
Svm+3-gram	0.257	0.375	**0.305**	0.173	**0.252**	**0.205**

表 7 不同方法在 sighan8 上的实验结果

（1）使用 3-gram 作为语言模型的计算方法要明显优于基于词/字向量的计算方法，而

两者主要的差别体现在准确率上。经过对一些实验结果进行分析后，我们发现利用上下文词向量计算句子得分的方法会受到分词错误的影响而降低计算结果的可靠性。另外，使用词向量本身作为语言模型的计算方法是否合理也仍值得商榷，因为在分析错误样例的时候，存在着很多原本正确的搭配被模型修改成了不正确的组合的情况。例如：原文是：“希望/天天/都/是/晴天”，这里面“都”和“是”被加入到我们的怀疑对象当中。然而经过词向量语言模型后修正的结果是“希望/天天/都/晒/晴天”。显然，“晒”和“晴天”对应词向量的相似度更高，因此按照我们做内积的方法得到句子的分数会增大。但这句话在语言上并不通顺，如果使用 n-gram 的话得到的分数就会明显低于原文。

（2）使用 SVM 作为判断错误位置在 SIGHAN8 上取得了很好的效果，但在 SIGHAN7 上的效果不佳，不如基于无监督的分词方法。这一点从数据集的分布上可以很明显地被发现。SVM 的训练数据采用的是 SIGHAN8 的训练集，与 SIGHAN8 测试集有着同样的数据分布。而其与 SIGHAN7 测试集的分布却稍有差别，这一点我们可以从表 5 的结果中看出。而这种数据分布上的差异很明显会影响错误位置的判断和后续的改正结果。

（3）Consec 模式的结果要优于 Single 模式的结果。Consec 是指从多个连续的单个字符至多改正一个字符，而 Single 是指可以改正任意多的字符。很明显，Single 模式考虑了更多的可能的错误位置，其检测和改正的召回率必然会有所提升。但对于准确率（precision）来讲，由于考虑了更多的可能的错误位置，其误报错误的概率也会提高，准确率会有所下降。另外我们发现，Single 模式会对于一个错位位置给出多个错误改正方法，导致准确率的大幅下降。例如：“会机极的克服” 在 Single 模式下被改为“会积极的克服”和“会机器的克服”两种结果，但在 Consec 模式下只会给出“会积极的克服”这一种正确的答案。总的来说 Consec 模式在准确率和召回率上的权衡要比 Single 模式做的更好。

4、样例分析：

从表 6,7 的结果我们可以看出，无论是错误位置检测还是错误改正，其准确率和召回率都不是特别高，尤其是准确率非常低，说明模型给出了一些错误的修改意见。那么下面我们就针对一些模型给出错误的修改结果进行分析，尝试找到模型存在的不足之处和值得改进的方向。

让/大家/公认/它/是/神木	“它”替换为“他”
玉不琢/不成器	“器”替换为“绩”
从/温暖的/被窝/爬起来/时	“时”替换为“是”

表 8 一些假阳性的错误样例

表 8 给出了一些模型误判断为错别字的情况。理论上讲，如果一个语言模型足够的好，那么实际上替换过后的句子应该没有原句的得分高。但是目前我们所使用的语言模型仅考虑了 3-gram 的情况，距离理想的语言模型还有一定的差距，我们下一步打算使用更加复杂的语言模型来减少这种假阳性情况发生的次数。

与其/自/暴/自/气/不如	“气”替换为“己”
迎接/每/一个/固/难	“固”替换为“个”
孙中山/的/转/记	“记”替换为“机”

表 9 一些检测出错误但未正确改正的例子

表 9 给出的是模型检测到错误但未改正成正确答案的情况。我们发现对于一个词的拼写错误很可能有多种改法能够提高其语言模型的结果，例如最后一个案例中“转记”改为“转机”和“传记”单看词来说都是合理的。但只有把其代入到原文之中我们才能判断哪一个是更合适的改法。第一个例子也是同样的道理，只有将“自暴自气”这几个字一起考虑我们才有可能将其改正为“自暴自弃”，而只考虑 3-gram 的话，“自暴自己”已经是目前能得到的最好的结果了。

另外模型对于有些错误是很难检测到的，例如“一直看录音机”应该改为“一直看录影机”，“小朋友每天活动的挤入”应该改为“小朋友每天活动的记录”等等。这些错误和传统的拼写错误不同，需要利用上下文的语义信息进行进一步的筛选和排查。这些错误的检测目前来看不能通过传统的 pipeline 模式进行分析，利用上下文相关的预训练模型（Elmo【8】，BERT【9】）很可能会得到更好的效果。

五、总结

我们组的课题是：设计实现一种汉语文字自动检查系统。为了实现这个系统，我们采用了两阶段的方法：找出错误的位置、计算出正确的替换字符。前者依据“错别字会带来异常的分词结果”，通过分词结果来搜索错误位置；后者在开源的 confusion set 内寻找合适的替换词，计算替换词的语言模型分数。

整个模型分为以下四个步骤：分词、识别错误位置、根据 confusion set 找到备选答案、根据语言模型找到得分最高的答案。我们用了两种方法来识别错误位置，一种是直接根据分

词结果判断，另一种是 SVM。直接分词法将分词过后被分成连续两个单个的字加入候选的错误字列表当中，此外，若分词结果中两个字或三个字的词不在给定词典中，那么我们也将其放入错误候选列表中。SVM 则根据一些特征，直接判断某个字是不是错字。我们用了两种语言模型来计算正确替换字符，词向量方法和 n-gram 方法。词向量通过计算每个词和上下文的相似度，来判断哪个词更有可能出现在上下文中，但是我们发现这种方法很容易将正确的词改错。我们采用的 3-gram 方法，比词向量的结果好，但是仍然存在一些问题。

六、引用

[1] Yih-Ru Wang and Yuan-Fu Liao (2015). Word Vector/Conditional Random Field-based Chinese Spelling Error Detection for SIGHAN-2015 Evaluation. Proceedings of the 8th SIGHAN Workshop on Chinese Language Processing (SIGHAN'15), Beijing, China, 30-31 July, 2015, pp. 46-49.

[2] Yih-Ru Wang, Yuan-Fu Liao, Yeh-Kuang Wu and Liang-Chun Chang (2013). Conditional Random Field-based Parser and Language Model for Traditional Chinese Spelling Checker. Proceedings of the seventh SIGHAN Workshop on Chinese Language Processing (SIGHAN-7), Nagoya, Japan, 14 October 2013, pp. 69-73.

[3] Weijian Xie, Peijie Huang*, Xinrui Zhang, Kaiduo Hong, Qiang Huang, Bingzhou Chen and Lei Huang (2015). Chinese Spelling Check System Based on N-gram Model. Proceedings of the 8th SIGHAN Workshop on Chinese Language Processing (SIGHAN'15), Beijing, China, 30-31 July, 2015, pp. 128-136.

[4] Xiaodong Liu and Fei Cheng and Yanyan Luo and Kevin Duh and Yuji Matsumoto (2013).A Hybrid Chinese Spelling Correction Using Language Model and Statistical Machine Translation with Reranking. Proceedings of the seventh SIGHAN Workshop on Chinese Language Processing (SIGHAN-7), Nagoya, Japan, 14 October 2013, pp. 54-58.

[5] Wang D, Song Y, Li J, et al. A Hybrid Approach to Automatic Corpus Generation for Chinese

Spelling Check[C]. empirical methods in natural language processing, 2018: 2517-2527.

[6] Lee L, Yu L, Lee K, et al. A Sentence Judgment System for Grammatical Error Detection[C]. international conference on computational linguistics, 2014: 67-70.

[7] Li S, Zhao Z, Hu R, et al. Analogical Reasoning on Chinese Morphological and Semantic Relations[J]. meeting of the association for computational linguistics, 2018: 138-143.

[8] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[J]. arXiv preprint arXiv:1802.05365, 2018.

[9] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.