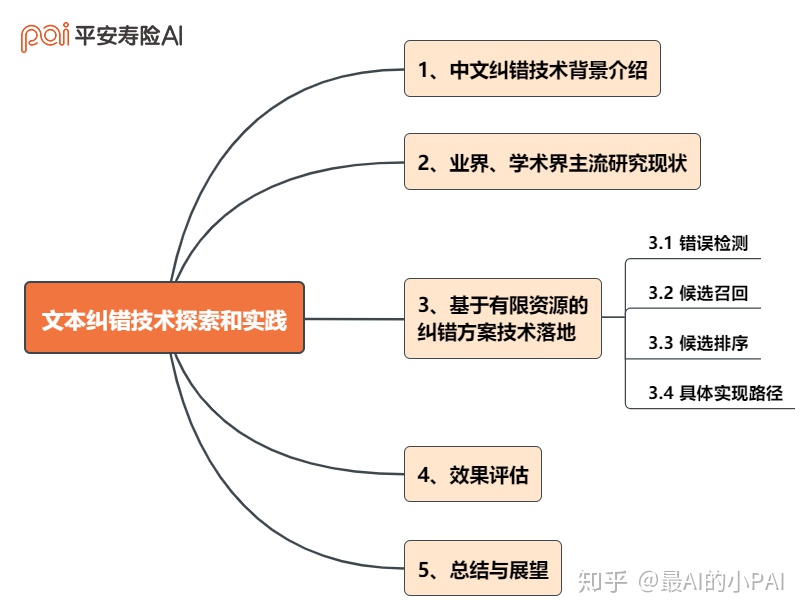
文本纠错技术探索和实践

# 1、全文框架概览



# 2、背景与意义

中文纠错技术是实现中文语句自动检查、自动纠错的一项重要技术，其目的是提高语言正确性的同时减少人工校验成本。纠错模块作为自然语言处理最基础的模块，其重要程度不言而喻。

在日常生活中，我们经常会在微信、微博等社交工具或公众号文章中发现许多错别字。我们在几个方面对文本出错概率进行了统计：在微博等新媒体领域中，文本出错概率在2%左右；在语音识别领域中，出错率最高可达8-10%；而在平安人寿问答领域中，用户提问出错率在去重后仍高达9%。

在平安人寿问答领域的用户问题中，我们发现多种类型错误。其中占比最高的错误是语言转化和发音不标准的错误，占错误总量的50%。比如一款保险产品“少儿平安福”被语言识别转化为“少儿平安符”、“飞机”因方言差异被读成“灰机”、“难受想哭”变成“难受香菇”等。

占比第二高的错误类型是拼写错误，占错误总量的35%。这些错误主要发生在通过拼音、五笔和手写输入文本的场景。比如“眼镜蛇”-“眼睛蛇”、“缺铁性贫血”-“缺铁性盆血”等。剩余的错误我们将其分类为语法和知识错误，语法错误包括多字少字乱序，如“地中海投保”-“投保地中海”，知识关联错误如“福田平安金融中心”错写为“南山平安金融中心”。

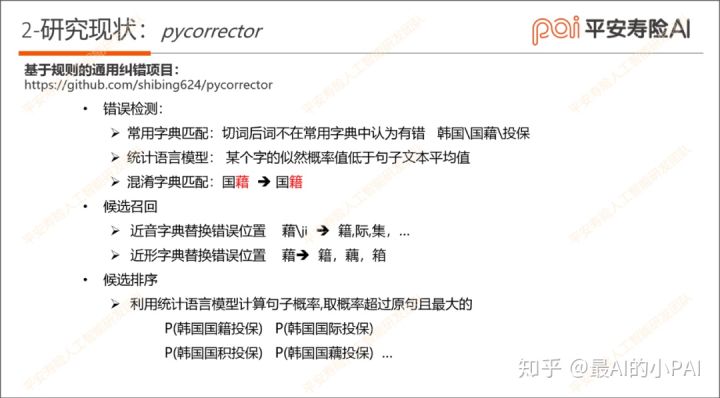
文本中大量的错误对上层nlp技术来说无疑是一项巨大挑战，输入数据的准确性是自然语言处理常见任务的基本前提，也是提高上层应用性能的关键。



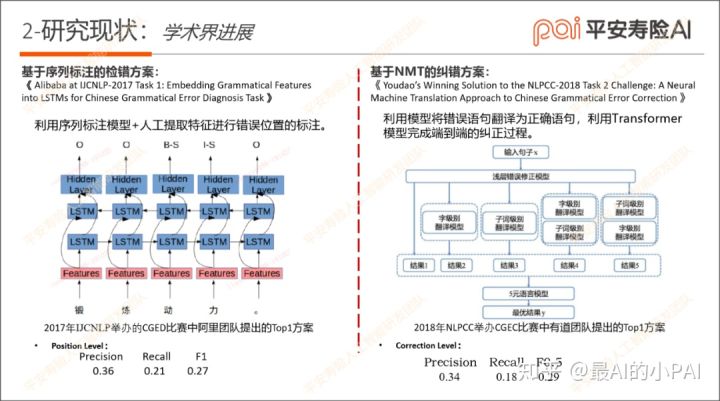
# 3、研究现状

在通用领域中，中文文本纠错问题是从互联网起始时就一直在解决的问题。在搜索引擎中，一个好的纠错系统能够对用户输入的查询词进行纠错提示，或直接展示正确答案。

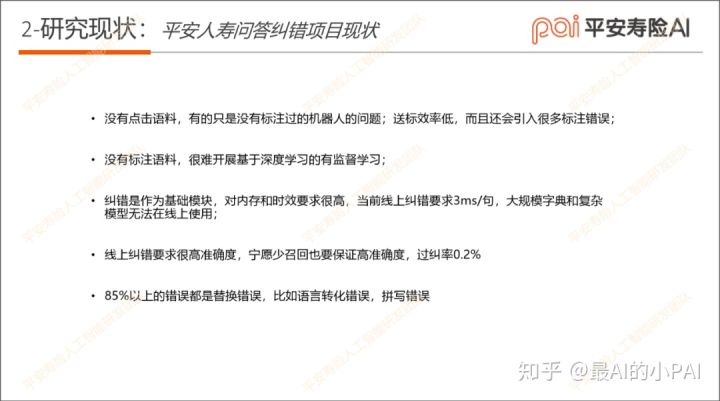
在此给大家介绍一个比较受欢迎的纠错项目：Pycorrector。该项目由规则纠错和深度学习纠错两部分组成。深度学习纠错项目中提到一些前沿的方法，比如机器翻译，但作者未提供直接调用接口；而规则纠错虽然可以直接调用，但因其性能和准确率无法满足我们项目需求，无法直接使用。下面简单介绍一下规则纠错，主要分为经典三步曲：第一步通过常用词词典匹配结合统计语言模型的方式进行错误检测；第二步利用近音字，近形字和混淆字进行候选召回；最后一步利用统计语言模型进行打分排序。



而中文文本纠错学术进展主要集中在比赛项目上，如前几年SIGHAN举办的中文拼写纠错比赛以及近几年NLPCC等举办的中文语法检测和纠错的比赛等。在2017年的中文语法错误检测比赛中，其Top1的主要方案利用了序列标注模型结合人工提取特征。在NLPCC2018年举办的中文语法纠错比赛中，冠军团队应用基于Transformer的翻译模型，其主要原理是将错误句子翻译为正确句子。

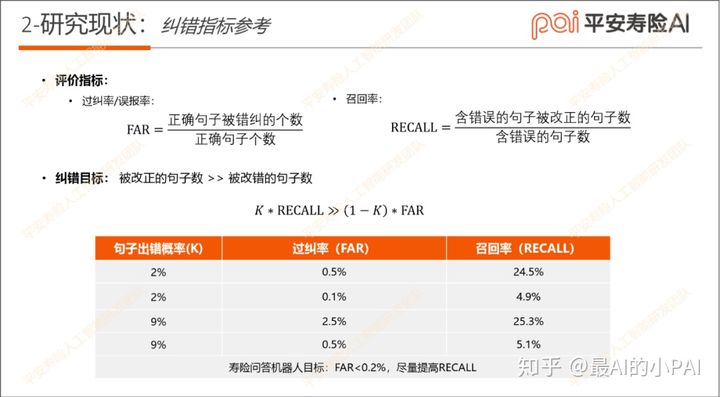


在学术界的进展中我们可以发现，很多成熟的方案都是基于有监督的深度学习，比如：序列标注模型、翻译模型。但在标注资源受限的情况下，此类深度学习模型很难应用落地。同时纠错技术本身作为文本预处理过程中的一环，其对线上时效性具有极其严格的要求，一般工业上的单句耗时要求在毫秒级别，因此极需高性能且高效的解决方案。



对于纠错系统的性能评估指标我们将过纠率与召回率看作硬性指标。过纠率代表正确的句子被改错的比率，召回率代表错误的句子被全部纠正的比率，其中较大的过纠率将会对系统和用户体验带来负面效果。我们的目标就是要让纠对句子数量远远大于被改错句子的数量，公式表述如下：如果句子出错概率是K，则K\*RECALL>>(1-K)\*FAR。

这里罗列了一些参考值，假设句子出错概率为2%，过纠率为0.5%，那么召回率必须要大于25%；如果我们句子的出错概率为9%，在同样的召回率情况下，我们可以容忍更大的过纠率。所以本系统的目标就是在控制过纠率在0.2%左右，尽量提高召回率。

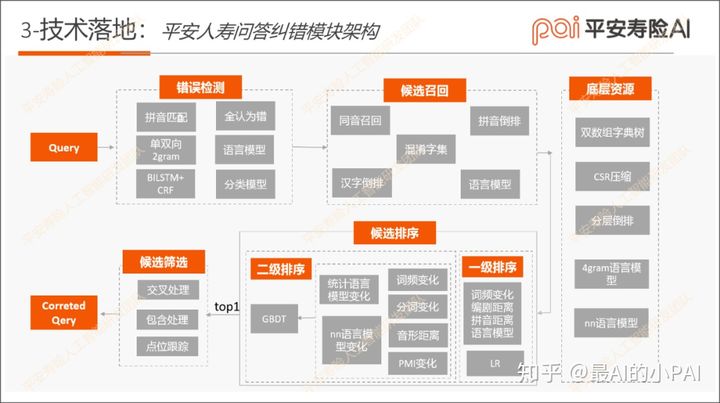


# 4、寿险垂域纠错技术落地

为了解决上述挑战，团队设计了纠错框架（如下图），在主流程上我们延用了业界经典的纠错系统架构：错误检测、候选召回、候选排序。这里为了更好展示我们纠错系统的细节，我们也将“底层资源”与“候选筛选”作为基础模块添加至主逻辑中。

在系统输入用户问题后，首先进入到错误检测模块找出相应的错误点位，然后针对错误片段进行正确候选词召回，其次经过粗排序、精排序等排序模块，最后通过候选筛选模块处理候选交叉冲突等情况。

其中底层资源用于存放各类底层字典资源、上下文语义信息从而方便各模块调用，其存储的数据结构主要包括：双数组字典树、稀疏矩阵压缩算法CSR和分层倒排索引等；上下文语义信息则以统计语言模型及NN语言模型的形式保存。



在主流程中，“错误检测、候选召回、候选排序”每一步我们都进行了多种解决方法的探索，下面将进行详细介绍。

## 4.1 错误检测

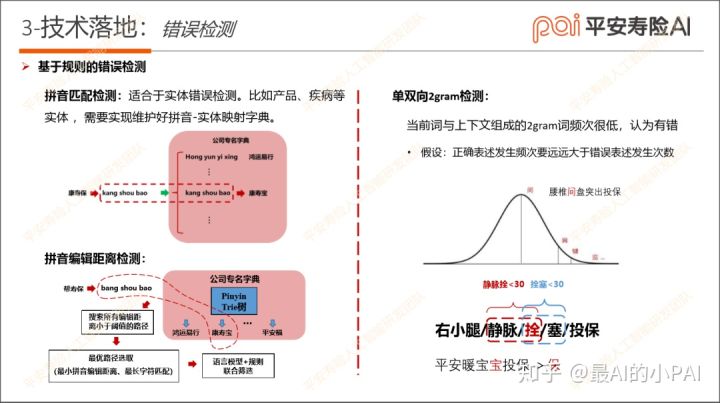
### 4.1.1 基于规则的错误检测

**1）基于近似拼音匹配的产品专名纠错**

在平安寿险的垂直领域，存在着多种产品，如：福保保，平安福。在通用的语言识别或者拼音输入法中，这些名称往往会被错打成福宝宝，平安符。此类实体错误多为近音错误，因此可建立拼音到实体的映射字典，从而完成错词到拼音再到实体的纠错流程。但是当错误片段的拼音不在映射表中时就无法进行匹配。因此我们将映射字典的存储结构改进为拼音Trie树，从而可以进行拼音编辑距离的匹配。

**2）双向2gram检测**

该方法使用我们基于一个假设：就是正确表述发生频次要比错误的表述发生频次要高很多。我们将语料中所有2gram的联合概率分布拟定为正态分布，正确2gram片段的出现概率将远大于错误出现的概率，从直观上来看这种假设也符合常理，因此可在此假设的基础上将联合概率的比值作为错误判断的依据。



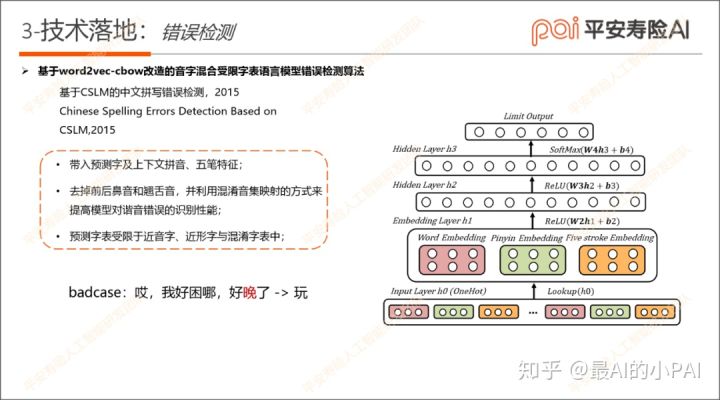
### 4.1.2 基于模型的错误检测

受到Bert的Mask Language Model的启发，我们可以把错误检测的过程转化成字级别的完形填空问题；在进行错误检测时对每个字所在位置进行逐一预测，从而根据预测的概率分布判断当前位置是否有错。

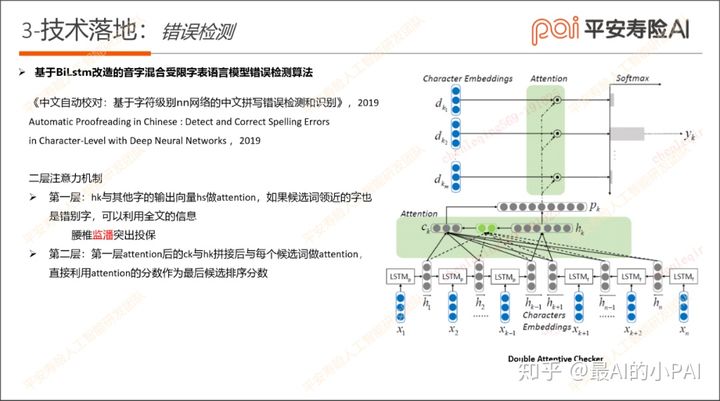


第一种语言模型的方法是基于word2vec的cbow改造。在word2vec训练过程中我可以通过指定窗口大小的上下文来预测当前字的概率。同时在此基础上，我们又进行两个方面的改造：

* 传统语言在预测当前位置时候是不会带入当前字的先验信息，但是在我们的场景中，因为正确字可能为错误字的近音、近形词，所以我们加入了待预测字的拼音和五笔特征；
* 我们将字典压缩为领域内的高频字，同时对NN的输出进行受限，在预测时输出层只计算近音、近形和混淆字所对应的神经元，在提高正区分度的同时，增加计算效率。



上文word2vec虽然能帮我们利用上下文的信息，但受预设窗口大小的限制，这些信息均是局部信息，长距离的依赖使用该模型无法解决。因此我们引入第二种语言模型的方法，该方法基于2019年的一篇论文进行改造的，模型利用了BiLstm，前向Lstm从左到右学习h\_k-1，后向Lstm从后到左学习h\_k+1，然后合并两个得到h\_k,得到h\_k先与输入的字向量x\_i做Attention得到C\_k,然后C\_k与h\_k拼接得到P\_k；再用P\_k与候选字向量做Attention，用Attention后的分数作为预测概率分布。使用该模型可以缓解邻近字也是错别字的情形；比如“腰椎键潘突出投保”，正确是“间盘”，我们在纠正“键”的时候，因为可以利用更长距离的“腰椎、突出、投保”等字符信息，可以减少右边错别字“潘”的影响。



第三种语言模型是基于Bert模型进行改造的，加入了拼音和字形特征，然后将训练的超参数进行调整，比如层数变成3层，embedding\_size从750降到150，使用5个头，字典从原来2w变成3.8k。

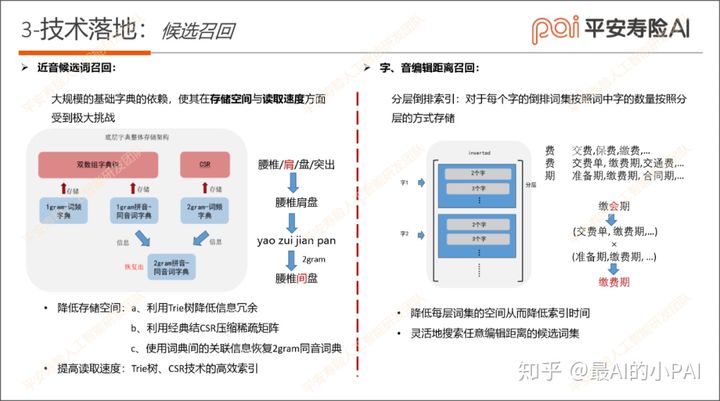


以上三个语言模型，考虑到效率问题，其中基于word2vec改造的模型用于线上纠错中；其他两个模型用于对效率要求较低，但准确度要求很高的离线版本的纠错中。

## 4.2 候选召回

上面介绍的语言模型除了用于错误检测，还可以用于候选召回，即正确的候选字从语言模型预测输出的topk中获取。

除了通过语言模型召回候选以外还需要通过混淆词典来召回正确候选词。相关的混淆词典包括：基于近音、近型、编辑距离的1、2gram混淆词典。为了提高字典的索引效率及搜索时间，我们将1gram词及词频和1gram近音词词典使用双数组字典树存储，而2gram词典采用CSR数据结构存储，同时2gram的近音混淆词可以从以上词典里恢复出来。而为了进行编辑距离召回候选词，我们建立了分层倒排索引词典从而提高搜索效率。

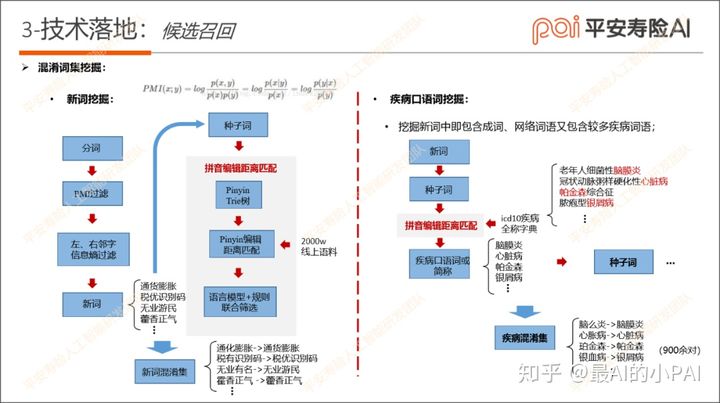


另外一种混淆词典的构建可以直接建立错误词与正确词的映射关系，为了保证较低的误纠率，其中的正确词需要是一些得到较高互信息分数的专有名词或成语，而错误词语一般为近音、近型的高频错词。

因此需要进行高频的混淆词挖掘。如下我们提供了两种混淆词挖掘方法：

方法一：首先采用新词挖掘算法挖掘语料中的新词，从而得到正确的种子词，再通过拼音编辑距离得到语料中与种子词拼音相近的错误词，从而通过人工标注得到高质量的高频混淆词典；

方法二：通过ICD10等标准的疾病词典与挖掘到的新词进行匹配，可以得到新词中与疾病相关的口语词，再根据疾病口语词挖掘与疾病相关的混淆词典。



## 4.3候选排序

对于一项纠错任务来说，其正确结果是具有唯一性的，那么就需要通过排序算法对召回的候选词进行打分排序，从而选择分数最高的一项进行替换。然而，候选召回模块得到的候选词数量庞大，逐一通过复杂模型计算替换概率将引入较大的时间损耗，因此在精排序前需要进行一定粗排序，从而通过简单的算法来过滤掉部分明显错误的答案，其中我们采用的是逻辑回归模型。人工抽取的特征主要包括：频率比值、编辑距离、拼音jaccard距离、Ngram统计语言模型分数差值等。



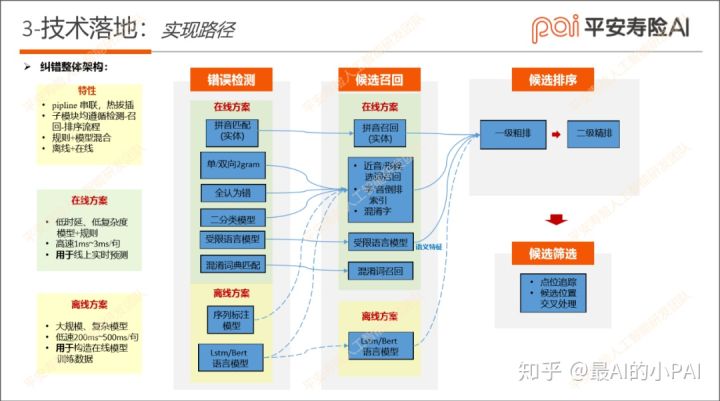
在精排序过程中我们采用Xgboost模型对候选进行打分，同时在特征工程方面我们也进行了更精细化地处理。候选特征主要分为局部特征、全局特征两个方面。局部特征主要包括：分词变化、频次变化、形音变化、PMI互信息变化、Ngram语言模型分数变化以及一些其他的基础特征。



## 4.4具体实现路径

下面我们对以上工作进行总结。整体的纠错框架可以分为“错误检测”、“候选召回”、“候选排序”三个主要流程，其中的每个流程我们都尝试了不同的方法，而不同方法间进行组合又可衍生出多种纠错方法。

在实际应用中，我们针对垂域中不同类型的错误采取了不同的纠错方法，方法间采用Pipeline串联的方式，使得不同的模块可以相互补充同时也方便模块线上的热拔插。根据算法的时效性我们将各类算法分为在线方案、离线方案两部分，其中离线方案具有较高的准确性但耗时较高，因此可通过离线方案进行原始数据去噪处理，来提升线上模型效果；而相对离线方案来说在线方案具有极高的时效性，线上平均每个Query的处理时间是1-3ms。



# 5、效果评估

线上我们分别对比了所提方案与Pycorrector的纠错性能。相较于Pycorrector，本系统的误报率为0.1%，召回率为70%，单句平均耗时在1.5ms左右，远高于开源项目。其中，系统仍存在一些badcase，如“好晚了”被误纠为“好玩了”等，此类badcase大都因为模型缺陷导致，其无法处理上下文语句、长依赖问题以及知识关联问题。



# 6、总结与展望

本系统的优点可以概括为以下几个方面：

* 方便扩展与领域迁移，对于新领域只需重新挖掘无监督数据即可；
* 系统架构方便拔插特殊编写的纠错子模块，方便后续的优化与开发。

同时也存在如下缺点：

* 难以应用于通用领域纠错；
* pipeline的机制导致错误逐级放大；
* pipeline同样导致串联链越长则耗时越长。

未来，当具有充足的标注语料时，需要强化上下文/全局语义理解，可训练深度学习模型提高系统召回性能；同时，基于神经机器翻译模型端到端纠错方法也是一种新的纠错思路。后续如保险垂域扩充了足够大的标注样本时，可尝试应用机器翻译算法优化整体纠错系统。另外，针对知识关联错误后续可尝试利用知识图谱完善整个纠错任务。