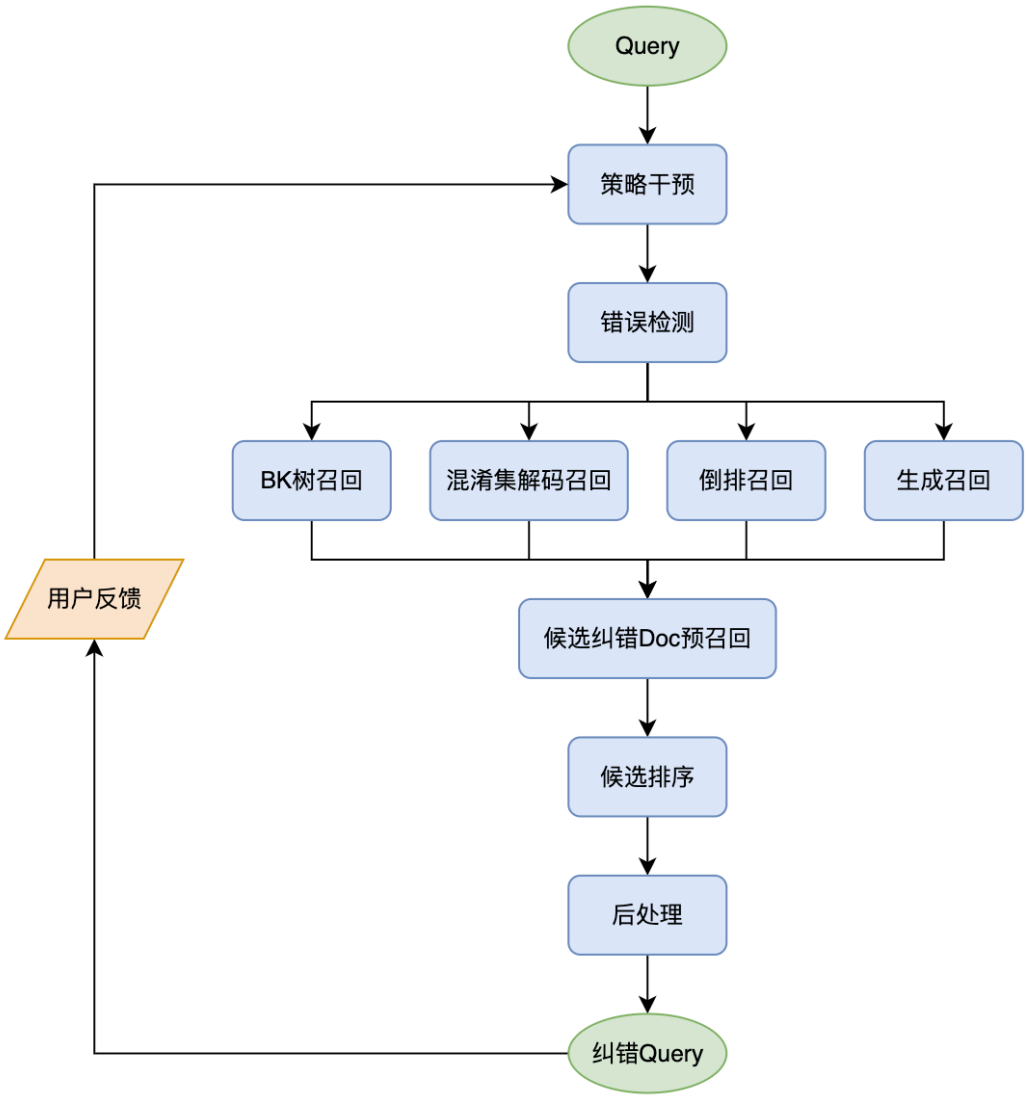
Query纠错

Query纠错本质上是Query改写中的子集，其主要针对用户输入中的拼写、语法、或格式错误，自动将错误的查询改正为正确的形式。通常来说，在通用搜索领域中，错误的Query在流量维度的占比大致在1%~2%左右，一些垂类搜索由于查询涉及领域专业术语，错误Query的流量占比可能会高达10%。虽然错误Query的占比较少，但是其对搜索质量、用户体验有较大影响，反映在业务指标上主要体现在：Query点击率、Query换词率、Doc零少结果率等。

Query纠错的产品形态和应用方式主要有：

- **直接纠错**
  - 当系统强置信判定Query错误时，直接对Query进行修正，搜索系统采用纠错后的Query进行检索，并给出可交互的提示词：“已为您显示`x`的搜索结果，仍然搜索`y`”，其中`x`为纠错后的Query，`y`为原检索词，用户仍可以点击原检索词`y`以显示`y`的搜索结果
- **纠错建议**
  - 系统提供可能的纠正建议供用户选择，用户可以直接接受建议进行搜索。与直接纠错不同，纠错建议让用户参与决策，允许用户决定是否采纳修正。系统此时仍然显示以原检索词Query进行检索后的结果，并给出可交互的提示词：”您要搜的是不是`x`“，其中`x`为建议的纠错Query，用户可以点击`x`获得以纠错Query`x`检索后的结果
- **纠错包含**
  - 系统显示的检索结果中不仅包含原检索Query的检索Doc，同时也包含纠错Query的检索Doc

主流的纠错架构采用pipeline的方式：**错误检测**、**候选召回**、**候选排序**。由于纠错模块通常处于搜索链路最上游，QP其他模块强依赖纠错结果，所以纠错模块需要尽可能的轻量化处理以保证时效性。另一方面，纠错由于其面对大量多样化的Corner Case，为了保证纠错效果需要做技术精细化处理造成纠错系统复杂度较高，在实际线上应用中充满各种挑战，在设计上需要在效果与成本上要有多方权衡。



评测指标

衡量纠错系统的技术指标如下：

- **FAR（误纠率）：** 正确Query被错纠的数量 / 正确Query总数
- **Precision（精确率）：** 错误Query被正确纠正的数量 / 触发纠错的数量
- **Recall（召回率）：** 错误Query被正确纠正的数量 / 错误Query总数

错误类型

常见的Query错误类型如下：

- **同音字错误：** 弹枇杷 --> 弹琵琶
- **近音字错误：** 砸汁机 --> 榨汁机
- **形近字错误：** 麻辣哈利 --> 麻辣蛤蜊
- **词序颠倒：** 工人智能 --> 人工智能
- **多字少字：** 汽车油 --> 汽车油耗
- **汉字拼音混合：** 考试报名fei --> 考试报名费

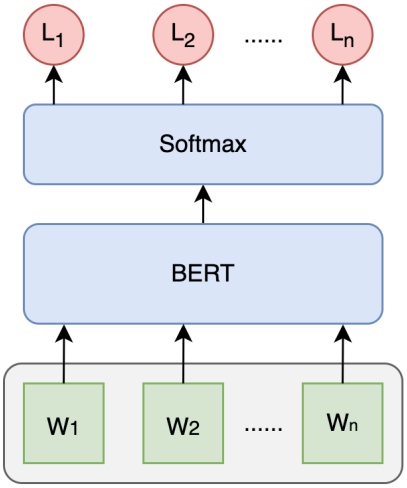
- **英文错误：** chargpt --> chatgpt

错误检测

Query纠错系统中一般需要检错模块用来判定当前Query是否错误，检错模块可以避免一些过纠、误纠的情况。另外由于线上真实错误Query的流量占比较低，而纠错系统的性能消耗通常较高，检错模块可以过滤掉大量无需纠错的搜索流量，避免给纠错系统带来性能压力。

常见的检错方式主要基于白名单知识库、历史搜索统计特征和语义信息。具体的，可以通过Query历史搜索统计特征（是否高频输入、高频点击、召回精确匹配Doc等）、百科数据知识库、用户反馈等信息构建白名单，命中白名单则判定无需纠错。为了保证纠错召回率，检错白名单只需保留强置信的无需纠错Query即可。

此外，还可以基于文本语义判断Query是否出错，业内常用的检错语言模型采用BERT + Softmax的方式对Query进行字粒度的序列化标注，即对每个字输出标签表示是否错误:



候选召回

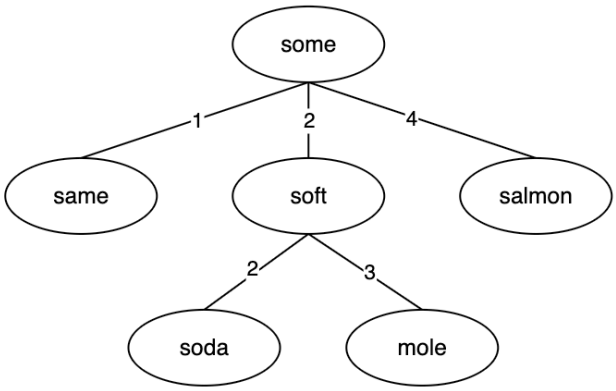
为了尽可能覆盖各种纠错类型，通常需要设计多路纠错候选召回。常见的纠错候选召回包括基于词表的BK树召回、基于混淆字词集的解码召回、基于字/拼音的倒排索引召回、基于BART的生成式召回。

BK树召回

BK树（Burkhard-Keller Tree）基于编辑距离（Levenshtein距离，包括插入、删除、替换），是一种用于高效查找相似元素的树形数据结构。

BK树的构建方式如下：

- **初始化根节点**：
  - 选择一个字符串作为根节点
- **插入新字符串**：
  - 对于每个待插入的字符串，计算其与当前节点字符串的编辑距离
  - 如果当前节点没有相同编辑距离的子节点，则新建一个节点；如果存在，则递归地在该子节点处进行插入



如上图：

1. 以some作为root节点
2. 插入same，与some编辑距离为1，建立新节点关联some，边权重为1
3. 插入soft，与some编辑距离为2，建立新节点关联some，边权重为2
4. 插入soda，与some编辑距离为2，递归插入到soft节点下，建立新节点关联soft，边权重为2
5. 插入mole，与some编辑距离为2，递归插入到soft节点下，建立新节点关联soft，边权重为3
6. 插入salmon，与some编辑距离为4，建立新节点关联salmon，边权重为4

BK树查询方式如下：

- **计算查询字符串的编辑距离**：
  - 计算查询字符串与当前节点字符串的编辑距离
- **递归查找**：
  - 如果当前节点与查询字符串的编辑距离在指定的容忍范围内，则将该节点视为匹配项
  - 递归搜索子树中与查询字符串编辑距离在允许范围内的所有节点。递归过程中，只有当子树的编辑距离与查询字符串的编辑距离相差在一定范围内时才会继续递归，否则跳过该分支
- **剪枝优化**：
  - 在搜索过程中，如果某个分支的编辑距离大于查询字符串与当前节点的编辑距离的容忍范围，则可以剪枝，跳过这个分支，从而提高查询效率

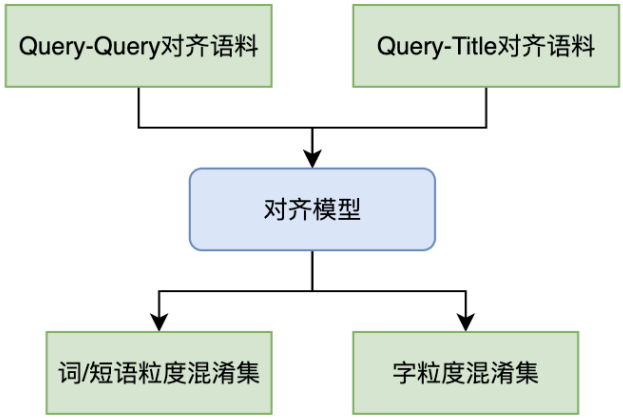
BK树的词表构建方式和错误检错中的白名单类似。在通过BK树获取纠错Query后，可基于统计特征变化（曝光/点击量）、Ngram语言模型概率、字形拼音编辑距离、PMI互信息变化等特征对候选结果过滤，并选取TopN作为BK树召回通道的结果。

混淆集解码召回

解码召回通过混淆字集等对Query中的字词进行替换获得纠错候选，并通过基于GBDT的排序模型选择满足阈值的TopN结果作为混淆集解码召回通道的结果。

字词混淆集构建

与Term改写的PT表构建方法类似，字词混淆集以Query-Title、共点Doc的Query-Query、用户Session内的Query-Query等做为对齐语料，通过对齐规则/模型，在同音、近音、形近等约束下挖掘得到多粒度（字、词、短语）的混淆集，混淆集格式为三元组<原词，转移词，转移特征>。



解码候选召回和排序

得到字粒度和词粒度的混淆集后，即可对Query纠错位置进行字/词替换构建候选纠错：

1. 对Query进行细粒度分词和字粒度切词
2. 可基于BERT检错模型序列标注对纠错位置进行限制
3. 对纠错位置的字/词从混淆集中召回候选，完成匹配替换
4. 对候选结果聚合、去重、排序，选择满足阈值的TopN的结果作为混淆集解码召回通道的结果

混淆集解码召回通路的排序模型通常采用轻量级的GBDT模型，常见的特征如下：

特征
Query长度/Term数以及对应特征变化（差值/比值）
错词位置/BERT检错模型序列标注打分
纠错类型
文本/拼音编辑距离
文本/拼音Jaccard距离
PMI互信息变化
Ngram语言模型困惑度变化
Query搜索统计特征及特征变化（搜索量/点击量/点击率/召回Doc量等）
纠错字/词转移特征（共现频次、左右熵、TF、IDF、IQF、相似度等）
候选来源

倒排索引召回

对白名单词表构建倒排索引：

- **创建候选词变体**：在索引中保存所有词汇的变体以实现模糊匹配，变体的实现通过通配符符号实现

    - `*`：表示任意字符（或一组字符，根据编辑距离限制）。例如，`appl*` 可以匹配 `apple`、`applle`、`appl` 等变体
    - `?`：表示单个字符的变体。例如，`a?ple` 可以匹配 `apple`、`ample`等变体
    - `^`：可以表示字符的顺序变动。例如，`n^ot`可以匹配`not`和`ont`
- **生成倒排索引**：对于每个词汇的所有变体，都需要为其创建倒排索引
- **查询纠错**：对查询词Query生成限定编辑距离内的变体，然后使用倒排索引召回相关文档，保留有效编辑距离较短的作为纠错候选召回

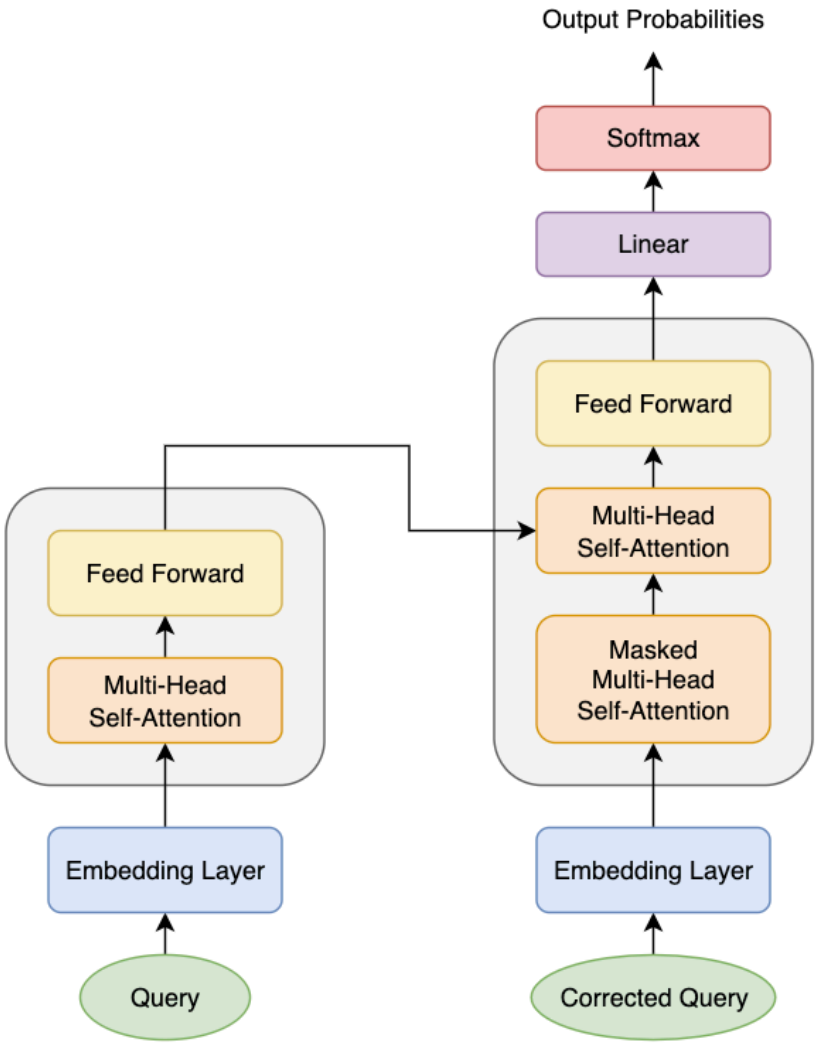
与BK树、混淆集召回相同，对倒排索引召回的纠错候选需经过后处理过滤筛选TopN纠错候选作为倒排索引召回通道结果。

BART生成式召回

传统的Pipeline方式迭代成本高，纠错能力受各模块制约，在算力充沛和延时支持的情况下，采用类似于BART的生成式纠错召回是最佳方案。BART的训练方法基于**去噪自编码器**（denoising autoencoder），这意味着它在处理损坏的文本和恢复缺失信息方面特别强大。对于Query文本纠错任务，BART可以很自然地处理输入文本中的错误并生成正确的文本。

BART

BART 是一个结合了**自编码器**和**自回归**模型的预训练模型，其编码器能够处理文本的上下文信息，而解码器生成每个输出词时依赖前面生成的词。



如图，左侧结构为Encoder，右侧结构为Decoder，Decoder中的遮蔽注意力层关注Decoder过去所有的输入，而第二个注意力层则是使用Encoder对整个句子的输出。

训练集构建

BART的训练数据以每一对（错误Query，正确Query）作为一条训练样本，错误Query为输入，正确Query为目标输出。

BART的纠错微调训练需要大量的训练样本，除去人工标注和线上挖掘数据，还可通过规则替换和语言模型大批量生成数据：

- 基于规则生成错误Query
    - 对正确Query通过混淆集字词替换、字符交换、增删字符等生成错误Query
- 基于语言模型生成错误Query
    - 利用微调后的T5、BART、GPT等生成模型对正确Query生成错误Query

对于构建的纠错数据，需要平衡正样本和负样本的比例，以及保证覆盖不同的纠错类型，并符合线上错误类型分布，同时避免生成的数据过于离谱。

候选排序

经过BK树、混淆集、倒排索引召回和BART生成获得纠错候选后，为进一步保证结果的准确率，需要对候选聚合打分，判定最终的纠错结果和纠错应用类型。

在候选纠错排序中，通过BERT模型从文本语义理解角度对纠错候选打分。此外，仅从Query维度实现检错和纠正通常是不足的，缺失了Query关联Doc的信息应用，通过对原检索Query和纠错Query进行预检索获得Doc信息可以有效辅助纠错决策。

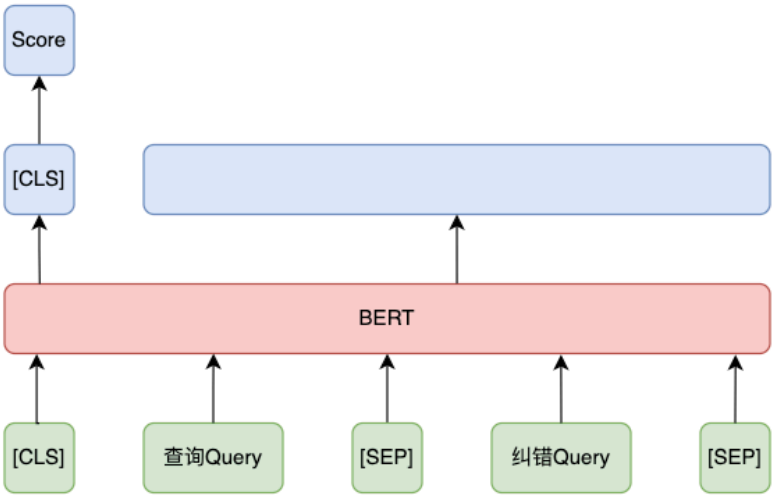
纠错预检索

实现纠错预检索实现流程：

- 构建一个轻量的索引库（通常采用倒排索引），索引库选择高质量的Doc（高曝光点击、时效性高、非广告营销等）
- 原始查询和纠错候选分别使用BM25模型进行检索，获取相关文档
- 基于检索的相关性文档信息，作为特征辅助优化纠错

BERT判别

BERT纠错排序输入为查询Query和纠错Query，输出纠错置信度分数。另外，在提高模型效果上，可以在训练样本中添加side information，比如Query关联的Doc信息作为上下文Context。



用户反馈采集和退场机制

纠错误纠会严重影响用户的搜索体验，用户反馈采集和退场机制是非常重要的组成部分。它们帮助系统不断改进和适应用户需求，确保系统能够动态调整和优化，提升用户体验和系统性能。

常见的反馈信息：

- **明确反馈**：用户明确表示是否接受纠错建议。例如，在纠错提示下用户点击交互行为
- **隐性反馈**：用户行为数据，如点击行为、查询行为、停留时间等，可以间接反映纠错的效果。例如，用户在收到纠错建议后是否继续点击相关结果，或者是否改变查询词
- **错误报告**：用户主动报告错误或不满意的纠错建议，通常通过按钮、留言、投诉等方式提交

当收集到一定规模的反馈数据可以用来作为白名单词库以及模型训练数据集。对于可能造成舆情的紧急的突发纠错问题需要触发退场机制及时干预问题Case，避免影响用户体验。

总结

Query纠错面对拼写多样性、复杂性性和实时性等挑战，涉及的错误类型种类繁多，不同类型的错误往往需要采取不同的纠错策略和技术。另一方面，纠错的目标比较明确，高质量的模型训练数据集很大程度上决定了纠错效果的上限。随着技术的发展，基于生成模型的纠错算法已成为主流，但也面临着算力成本和实时性等挑战。在实际应用中，需要根据具体场景和需求选择合适的技术方案。

参考文献

1. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
2. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension