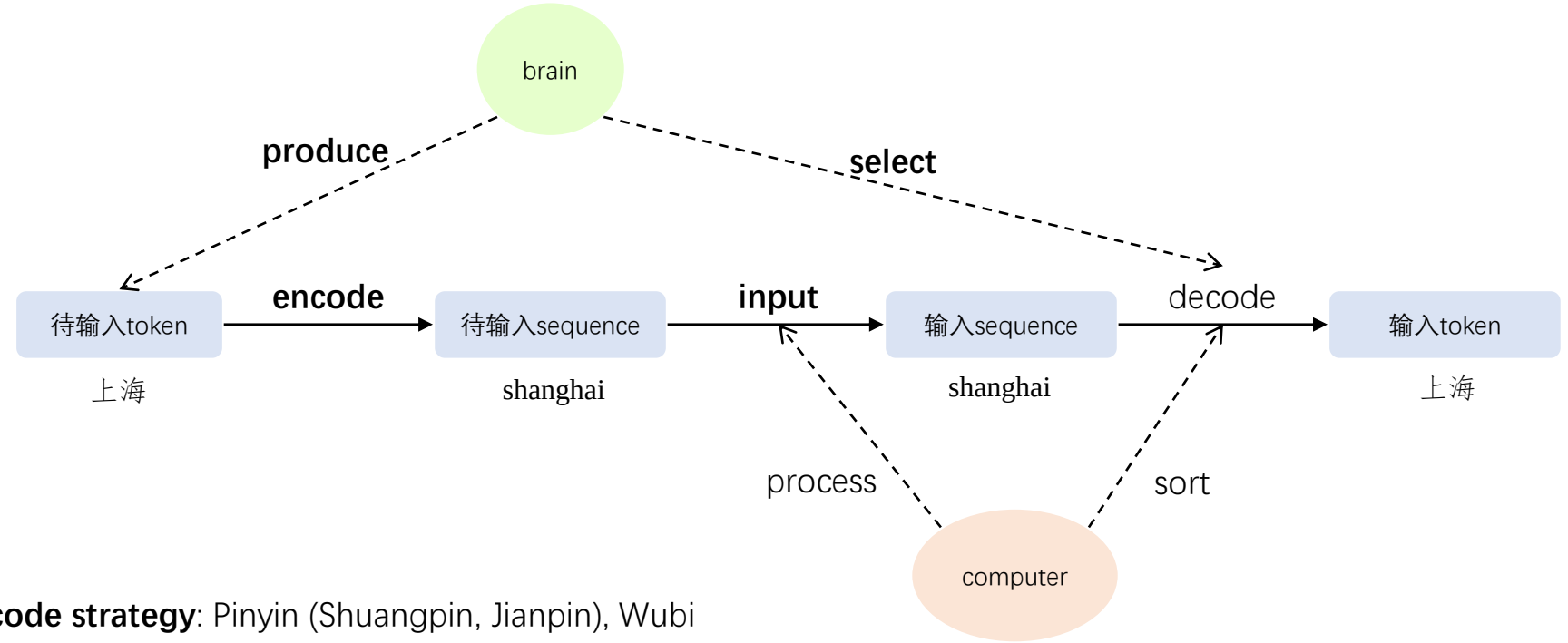
Confusion Set Generation

Chaoxu Pang

2021.7.29 - 2021.8.25

Cause and effect

Only human actions bring errors.



Encode strategy: Pinyin (Shuangpin, Jianpin), Wubi

Input strategy: 26 keys, 9 keys, handwriting

Cause and effect

Produce: 使用贬义词或屏蔽词

Encode: **模糊音（音节混淆）**, 输入序列缺省

Input: **键盘的输入误触**

Select: **同音词**

Edit distance algorithm

编辑距离：由一个序列转换为另一个序列所需要的最少单字符编辑操作次数

插入（Insertion）

删除（Deletion）

替换（Substitution）

	0	x	y	z
0	0	1	2	3
x	1	0	1	2
x	2	1	1	2
c	3	2	2	2

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

删除 a_i

插入 b_j

替换 b_j

将**次数**替换为**依赖于字符的得分**

Refined edit distance

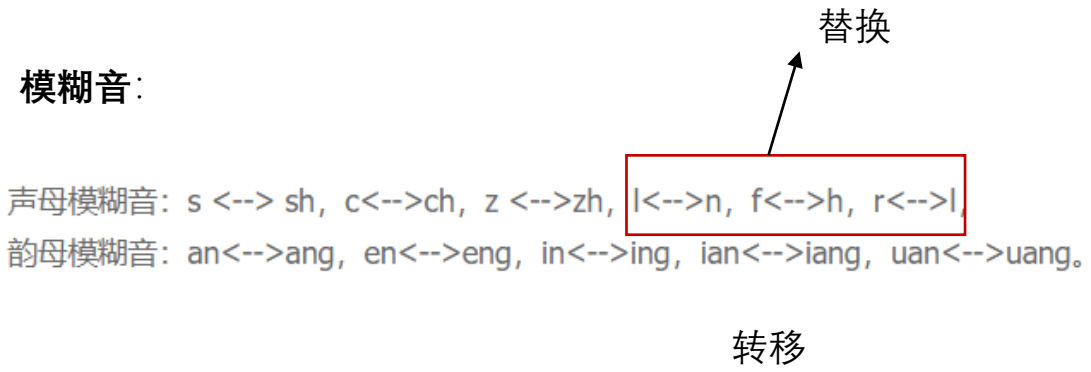
替换矩阵 S

S_ij 建模字母i被替换为字母j的得分。

转移矩阵 T

T_ij 建模字母j之后字母i被删除或插入的得分。

模糊音:



键盘输入误触:

替换　　w被替换成q, e　r被替换成e, t 等

每一类错误对应一个得分。模糊音（0.5），输入误触（0.25）

Example

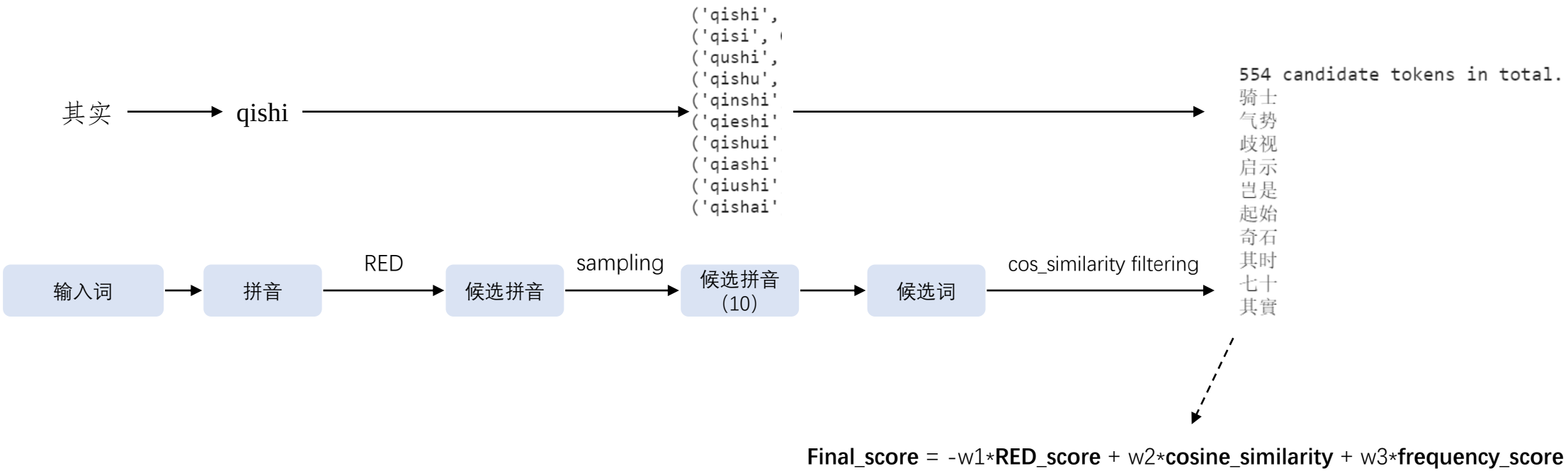
```
edit_distance('shanghai', 'sanghao')
```

```
[[0, 1, 2, 3, 4, 5, 6, 7],
 [1, 0, 1, 2, 3, 4, 5, 6],
 [2, 1, 1, 2, 3, 3, 4, 5],
 [3, 2, 1, 2, 3, 4, 3, 4],
 [4, 3, 2, 1, 2, 3, 4, 4],
 [5, 4, 3, 2, 1, 2, 3, 4],
 [6, 5, 4, 3, 2, 1, 2, 3],
 [7, 6, 5, 4, 3, 2, 1, 2],
 [8, 7, 6, 5, 4, 3, 2, 2]]
```

```
refined_edit_distance('shanghai', 'sanghao', score_matrix)
```

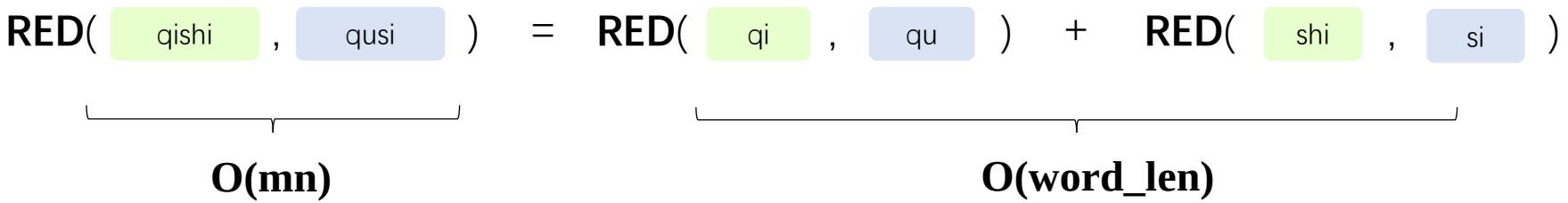
```
[[0, 1, 2, 3, 4, 5, 6, 7],
 [1, 0, 1, 2, 2.5, 3.5, 4.5, 5.5],
 [2, 0.5, 1, 2, 2.5, 2.5, 3.5, 4.5],
 [3, 1.5, 0.5, 1.5, 2.0, 3.0, 2.5, 3.5],
 [4, 2.5, 1.5, 0.5, 1.0, 2.0, 3.0, 3.5],
 [5, 3.0, 2.0, 1.0, 0.5, 1.5, 2.5, 3.5],
 [6, 4.0, 3.0, 2.0, 1.5, 0.5, 1.5, 2.5],
 [7, 5.0, 4.0, 3.0, 2.5, 1.5, 0.5, 1.5],
 [8, 6.0, 5.0, 4.0, 3.5, 2.5, 1.5, 1.25]]
```

Confusion set generation



RED score computing

两个词的RED分数可以视为各个字的RED得分之和。



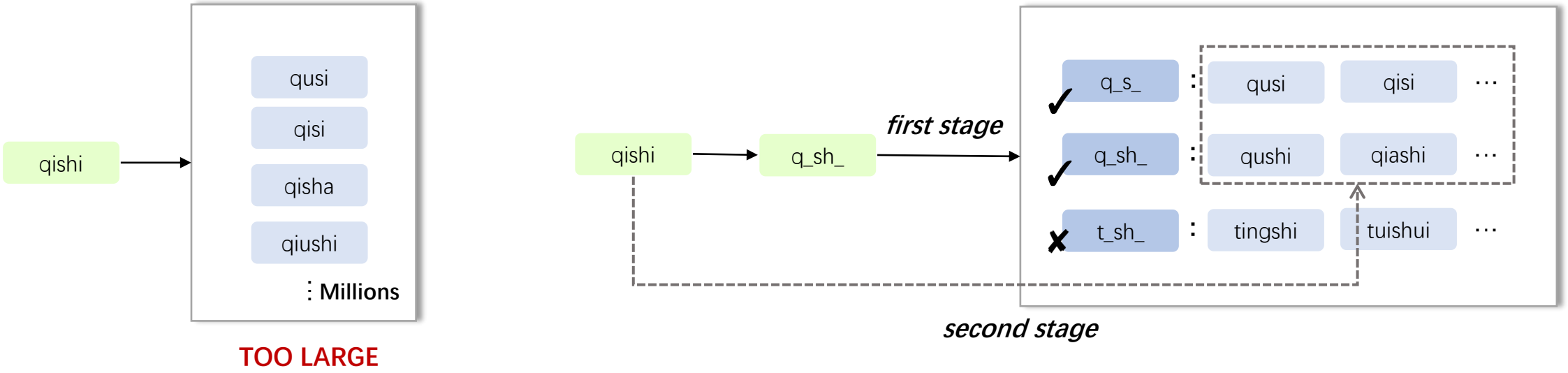
不同汉字拼音之间的RED可以提前计算得到，因此可以将计算复杂度简化为O(word_len)

一个词的平均时间由100s缩短到4s

Two-stage retrieval

第一阶段先进行声母序列的检索，得到一个较小的候选拼音序列集合。

第二阶段在得到的候选拼音序列集合上检索。



Embedding storage

当embedding文件很大时，内存条件不允许的情况下，将其拆解为尽量多的小文件进行存储。

本项目中使用含8M个词的200维向量，大小为16GB。

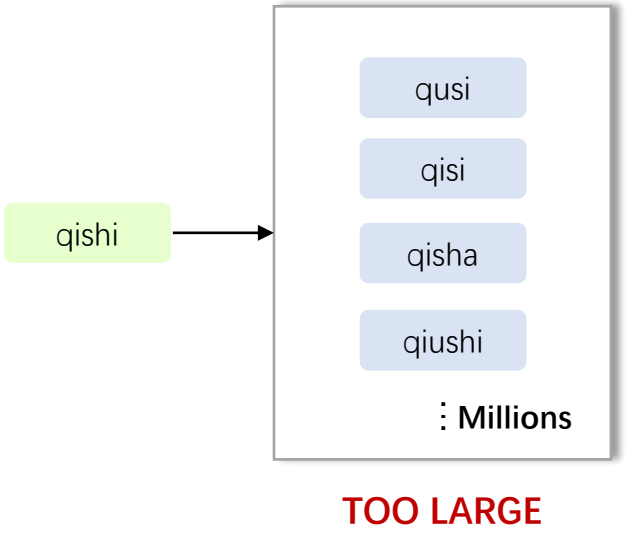
存储目录结构：embeddings/**word_length**/**first_char_pinyin**/**second_char_pinyin**.pkl

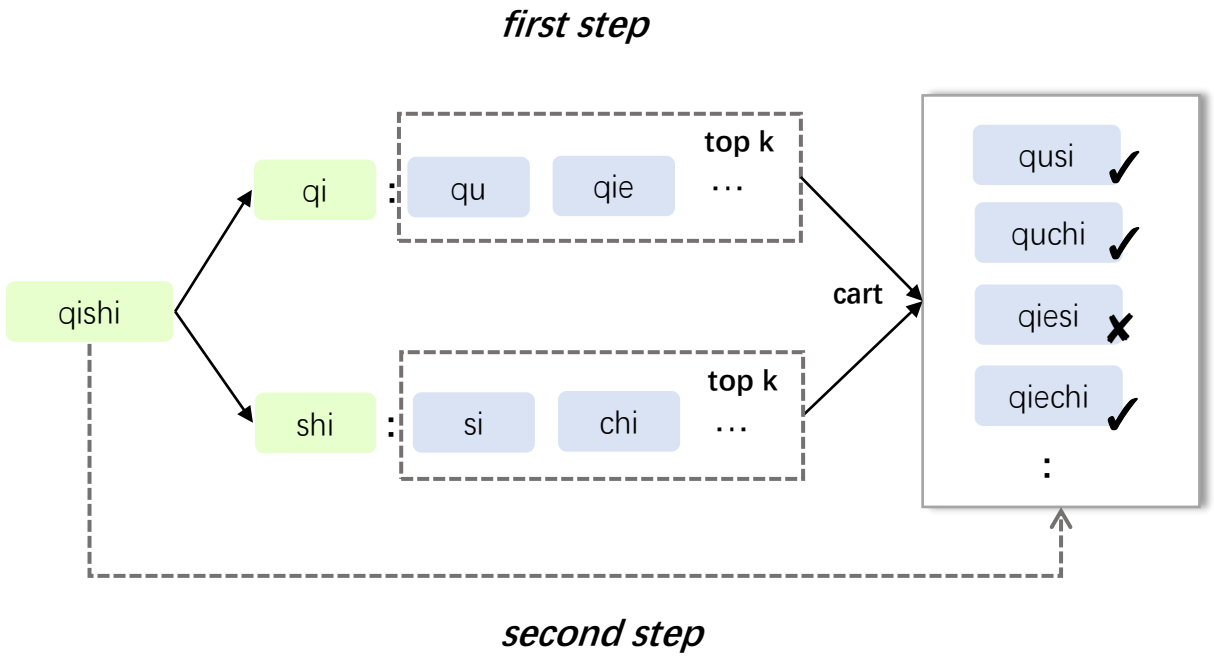
例如：“超短期”所在文件为：embeddings/**3**/**chao**/**duan**.pkl

DCC retrieval

第一阶段先得到每个字拼音的相似字拼音，计算笛卡尔积得到候选拼音序列集合。

第二阶段在得到的候选拼音序列集合上检索。

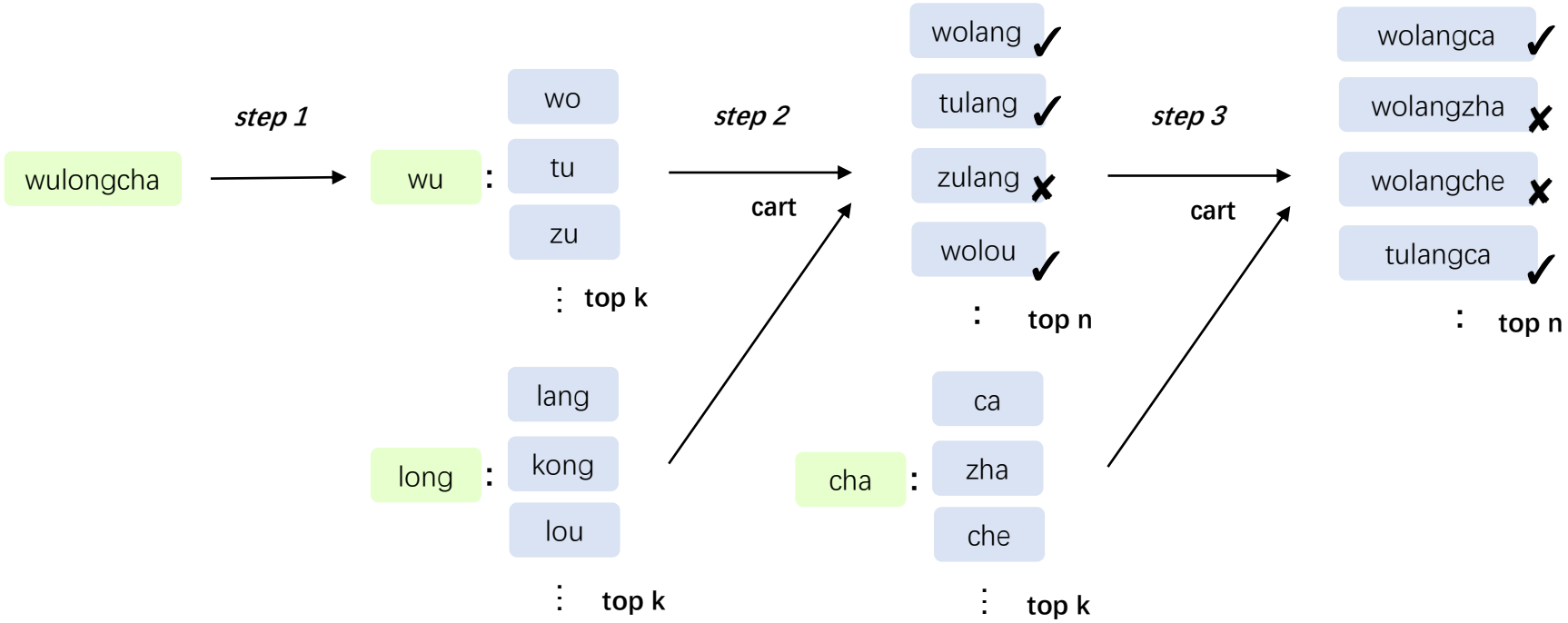




Beam search retrieval

迭代：在每一步首先得到每个字拼音的相似字拼音

然后与前步骤得到的候选拼音序列组合选出新的候选拼音序列



Recall evaluation

Normalized Cumulative Gain (NCG)

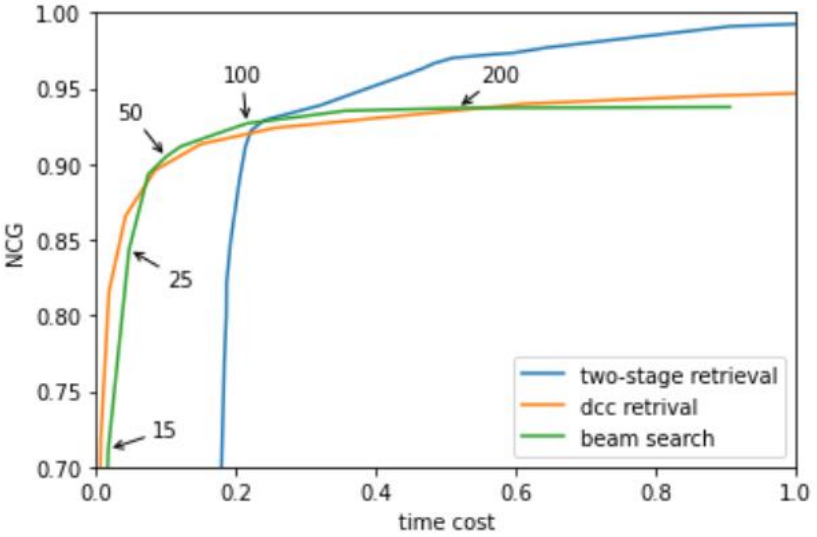
评估上述检索策略是否会漏掉重要的拼音序列。

借鉴NDCG，注意到RED产生的分数存在较多的并列，

采用按分数卡阈值而不是recall@N。

每个得分对应一个权重，对于真实检索结果中的前n个得分：

$$NCG@n = \frac{\sum_{i=1}^{n} w_i L_i^{retr}}{\sum_{i=1}^{n} w_i L_i^{truth}}$$



Word frequency score

Word frequency normalization

$$f^*(w) = (\frac{f(w) - f_{min}}{f_{max} - f_{min}})^{\alpha} \in [0, 1]$$

词频数据为包含66万词的金融文档词频统结果。

由于词频的幂律分布，引入指数因子对其进行平滑。本项目中取0.25。

对于不在统计结果内的词，其词频得分为0。

增大该得分的权重，模型将更倾向于输出常见词。

Sampling strategy

采样包括对**候选拼音**的采样和对**候选词**的采样。

sort：根据得分结果取前k个。

random：根据得分结果取前m个，再从剩下的结果随机取n个。（采样分布为得分的softmax）。

对于候选拼音采样，设置**special**模式以采样某种特定结构的拼音序列。

该特定结构可以通过观察数据归纳得到。