# 基于反馈的Query改写：你说过的，我才最懂

原创 玉琊 二二零号 2020-12-06

## 一、前言

本文对之前做过一段时间的Query改写（纠错，本文不严格区分这两种叫法）做一些总结，算法原理可以参考亚马逊的这篇论文：Feedback-Based Self-Learning in Large-Scale Conversational AI Agents。

## 二、方法

以前做Query纠错的一些通用思路是：基于大规模的线上日志训练一个相对置信的语言模型，基于相似度、编辑距离等方式挖掘一批高频词汇改写对，譬如说对于“伴奏兄弟”-->“半吨兄弟”这么一个改写对，原始ASR识别后的query有可能是：“播放伴奏兄弟的歌”，在经过中控的改写模块时，进行n-gram替换，按照语言模型的打分，发现：

> Score(播放 伴奏 兄弟 的 歌)> Score(播放 半吨 兄弟 的 歌)

并且分数值满足一定的阈值设定，那么可以把改写的query和原始query一起送入下游意图识别模块，看召回情况，再打分。

这种改写方式，有理有据，但相应的短板也非常明显：

- 高精度，但召回相对欠佳
- 改写词汇对维护成本高
- 意图打分模块，策略较重

另外还有一些seq2seq系列的方法，也做过相应的实验，这类改写方法过于不可控，当做玩具试试是可以的，包括也有Paper结合Bert来做的。

那么，有没有方法可以弥补上述经典纠错方法的短板呢？

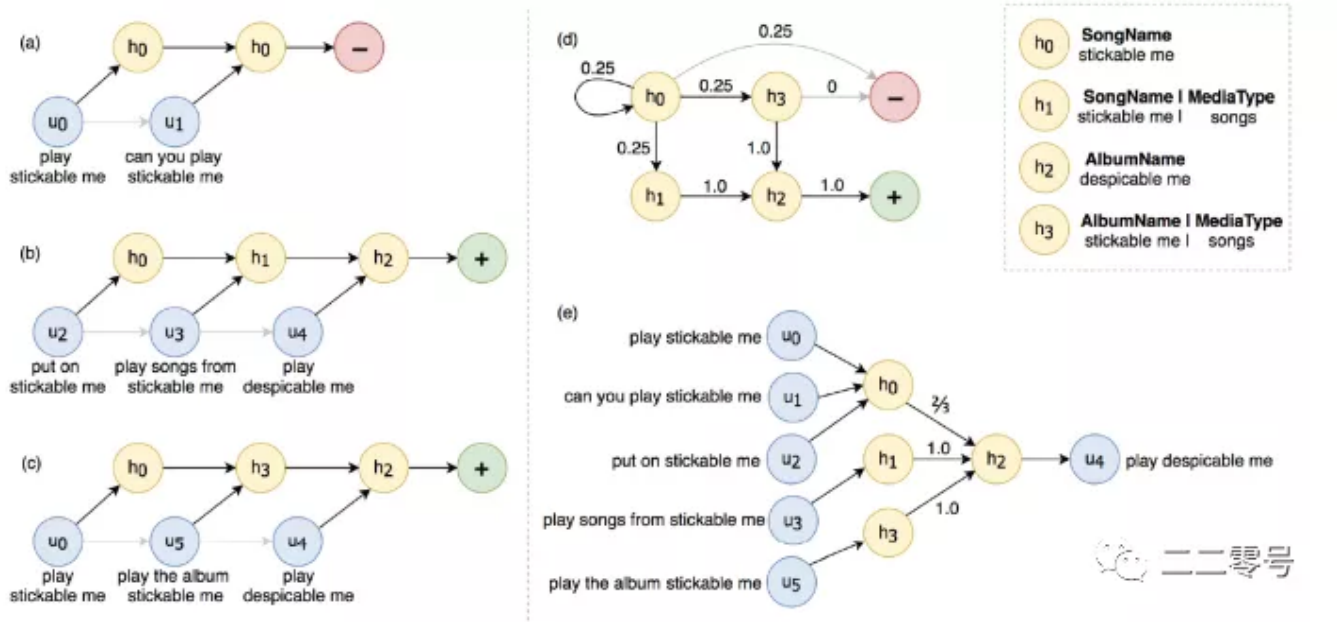
亚马逊的这篇Paper，提供了一种新的思路，笔者之前也已复现并小幅改进了该篇论文，效果属实惊艳。Paper干的事情，其实就是提出了一种挖掘Query改写对的方法，基于用户和chatbot的历史交互数据，挖掘改写对，并把改写对提供给线上改写模块，进行整句替换，像Alexa、小爱、天猫精灵等，都有着对应的应用场景。

**那么，Paper是怎么干的？流程如下：**

1. 用户query->dis映射
2. Session数据构造

3. Markov离线计算
4. 最优NLU修正计算
5. 融合音乐领域用户播放行为&人工Review

主要讲讲前三个步骤，我们分别来看，以用户和chatbot交互的历史数据为例，第一步干的事情，是把这些历史数据构造成query->dis的映射结构，关于dis是啥，来看paper中一张图：




utterance、dis转移示例

上图中的hidden state组成了interpretation space，在实操中，笔者并没有直接把query对齐paper中的hidden state构建($h_0$,$h_1$,$h_2$)，而是将query映射到一个新的dis空间，dis由**domain+intent+slot**组成，我们要把相似的query给映射到统一的dis空间中。

第二步session数据，是由dis和reply组成的。那么session数据是怎么构造的呢，paper中做了一个约定，用户和chatbot连续两次的交互时间差在45s以内的，才能属于同一个连续的session。以上图c为例，下面是user和chatbot的一次完整交互记录：

> user：play stickable me
>
> chatbot：？？？
>
> user：play the album stickable me
>
> chatbot：？？？
>
> user：play despicable me
>
> chatbot：ok.

前两轮的交互中，由于ASR识别错误、用户表述不明等可能的原因，chatbot并没能正确识别用户意图， 前两轮user的utterance(query)分别被映射入不同的dis空间中，分别为：

1. music,#,music.dialog.general.general,#song:stickable me
2. music,#,music.dialog.general.general,#album:stickable me

最终都未能成功播放，最后一次成功了，映射入成功的dis空间：

music,#,music.dialog.general.general,#album:stickable me

当我们构建好了足够多这样的session数据后，就能成功地计算出不同的utterance和dis之间的转移概率：

$$P(h|u) = \frac{c(u,h)}{\sum_{h' \in H} c(u,h')} \qquad P(u|h) = \frac{c(u,h)}{\sum_{u' \in U} c(u',h)}$$

utterance-->dis, dis-->utterance计算

第三步，当session数据构造完毕后，我们便可以得到一个完整的dis空间，该dis空间各个dis之间的转移状态 也就能得到了。

得到了$P(H|U)$,$P(U|H)$,$P(H|H)$后，回过头来想想问题是不是可以转换为：当由于ASR识别错误或者用户表述错误等原因，chatbot未能正确识别用户的意图时，我们需要计算出一条最有可能的路径，以到达错误utterance最有可能的改写utterance？而第一步，即要求出原始错误的dis到达最有可能的dis的路径。此时，我们引入吸收马尔科夫链，关于吸收马尔科夫链的具体介绍，可以自行搜索学习。

为计算出每个可能的非吸收态到吸收态的概率，构造转移矩阵如下：

$$\mathbf{A} = \begin{bmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I}_2 \end{bmatrix}$$

这里的$Q$是dis之间的转移矩阵，$R$是各个是各个dis状态到成功/失败状态的转移概率矩阵，$I_2$是一个单位矩阵。

中间还会有几步矩阵的计算比较关键，这里不细讲，最终我们会得到一个从起始dis状态到达success状态的最有可能路径，结合前述得到的$P(H|U)$和$P(U|H)$，我们即能算得最有可能的源utterance到目标utterance。

$$u_t^* = \arg\max_{u_t} \sum_{h_s} \sum_{h_t} P^{(1)}(u_t|h_t) \cdot \Phi_\infty(h_t) \cdot P^{(1)}(h_s|u_s)$$



目标utterance的计算

当然，仅仅是得到了改写的utterance对还不行，一般还会有一些后处理操作，对应流程中的第四第五步，包括结合用户播放时长、切歌率等特征的后处理，以及会有上线后的改写utterance跟踪打点模块。

## 三、结果

下面来看几个改写例子，有一个直观的感受：

- **金来**-->播放**惊雷**
- 鸟儿对**话**说-->鸟儿对**花**说
- 给我打了羊-->播放你的酒馆对我打了烊
- 泰国新加坡-->咖喱咖喱
- 闭上你的狗嘴-->暂停

第一个改写对，**金来**改写为**惊雷**，《惊雷》是2020年的一首神曲（？），在挖掘的过程中，ASR误识别为”金来“，我们将其正确改写为了惊雷，这个case用编辑距离、相似度结合打分的方法也能解，第二个case同理；

同样，在和chatbot交互的过程中，可能存在用户表述不完整或者由于噪声影响A导致ASR识别不全等情景，以第三个case为例，便可给出算法认为改写正确的utterance，算是一种信息的补全；

第四个case说明的则是另一种比较特别的改写场景，“泰国新加坡”是歌曲《咖喱咖喱》中的一句歌词，用户在和chatbot说“泰国新加坡”的时候，chatbot搜索相关资源，在我们的场景下，一般会落到百科的domain，返回“泰国新加坡”的百科解释，但其实我们是想听《咖喱咖喱》这首歌，那么当chatbot返回用户不想要的资源时，用户会频繁地打断chatbot，甚至到最后就索性让chatbot“闭嘴”了，对应落到**failure**的dis状态，而那些最后正确说出《咖喱咖喱》的歌名的用户，则会允许chatbot播放完这首歌，对应落入**success**的dis状态；

第五个case，是用户想要让chatbot中止当前的工作，用比较口语化的语言让chatbot终止状态，然而chatbot无法理解，只有在用户说了比较清晰直观的祈使命令，chatbot才会执行对应操作。

实操中，在仔细分析了大量的改写case后，我们认为类似于第四、五种case，已经不是传统意义上的改写任务，而更加是一种推荐任务。

## 四、结尾

以上是基于Amazon的Paper以及自己的一些实操经历，针对多轮对话类改写任务的一点小小总结，在工程上，对于上述的方法也做了一些尝试，非常有趣。此外，在面向头部和长尾的query改写任务时，试过不少的方法，也踩了不少的坑，欢迎相关研究方向的小伙伴一起入坑讨论。

## 五、参考文献

[1] Ponnusamy, Pragaash, et al. "Feedback-based self-learning in large-scale conversational ai agents." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 08. 2020.