特型卡片搜索系统-清华合作课程项目

一、项目基本情况

国内/外各大搜索引擎的综合频道除了满足自然结果检索和广告需求外，还需要进行特型卡片的承载（如：头条搜索中的“如意”；百度搜索的“阿拉丁”；以及360搜索的onebox等）。实现效果如图（下方为三个特型卡片的搜索示例）：

就这类特型卡片的承载，一般具备以下特性：

1. 产品形态的优势：富交互；缩短需求满足的路径（更好的摘要满足）
2. 被检索数据：高度结构化，一般是由第三方合作方提取而来的优质数据。数据质量高且工整。
3. 覆盖类目/场景众多（几百计，且未收敛）。不同卡片的召回方案异构性强，很难像网页搜索场景，构建一套非常general的搜索解决方案
4. 综合搜索场景下，由于用户没有明确的先验意图（区别于：番茄小说app下的搜索一般是消费小说阅读需求等）。特定类目的特性卡片的召回过程，往往对召回准确率的要求远大于召回率（误召回对搜索引擎用户体验的伤害大于欠召回），这也就需要按类目做召回方案的精耕细作。

真实业务场景分析

头条搜索中，“车型价格-如意”卡片。q=宝马ix3的价格是多少。线上召回效果：

那么，实际上的线上检索系统是如何做到的呢？

1. 原始的结构化数据（demo）

由“懂车帝”生产，所得的结构化数据。

JSON


```json
{
  "brand_name": "宝马",
  "cars": {
    "item": [
      {
        "car_id": "53947",
        "car_name": "2021款 改款 领先型",
        "car_sale_status": "在售",
        "price_presale": "47.00",
      },
      {
        "car_id": "53948",
        "car_name": "2021款 改款 创领型",
        "car_sale_status": "在售",
        "price": "35.59",
      }
    ]
  },
  "series_id": "4538",
  "series_name": "宝马iX3",
  "series_sale_status": "在售",
  "series_type": "中型SUV",
  "sub_brand_name": "华晨宝马",
  "title": "宝马iX3",
}
```


下方为一条XML格式的示例。

XML

```xml
<DOCUMENT>
  <item>
    <display>
      <title>东风裕隆新能源</title>
      <pc_brand_url>https://www.dongchedi.com/ </pc_brand_url>
      <brand_url>https://www.dongchedi.com/ </brand_url>
      <sub_brands>
        <item>
          <tag>轿车</tag>
          <sub_brand_name>东风裕隆新能源</sub_brand_name>
          <series_list>
            <item>
              <pc_pic_url>https://www.dongchedi.com/ </pc_pic_url>
              <series_brand>东风裕隆</series_brand>
            </item>
          </series_list>
        </item>
      </sub_brands>
    </display>
    <key>1311东风裕隆新能源</key>
  </item>
  <item>
    <display>
      <title>宝马x3</title>
      <pc_brand_url>https://www.dongchedi.com/ </pc_brand_url>
      <brand_url>https://www.dongchedi.com/ </brand_url>
      <sub_brands>
        <item>
          <tag>SUV</tag>
          <sub_brand_name>宝马x3</sub_brand_name>
          <series_list>
            <item>
              <pc_pic_url>https://www.dongchedi.com/ </pc_pic_url>
              <series_brand>宝马</series_brand>
            </item>
          </series_list>
        </item>
      </sub_brands>
    </display>
    <key>1312宝马x3</key>
  </item>
</DOCUMENT>
```

2. 离线数据处理环节

1）抽取demo中原始数据中的“title”字段，并形成一个本地词典，用作后续的在线检索时使用

2）将demo中原始数据中“cars.item”数组中的数据拆分，并写入倒排引擎，供在线检索时使用

3) 将demo中原始数据中“cars.item”数组中的数据拆分，并写入摘要服务（kv类型），供在线检索时使用

此处的倒排引擎、摘要服务等会在后面具体介绍。

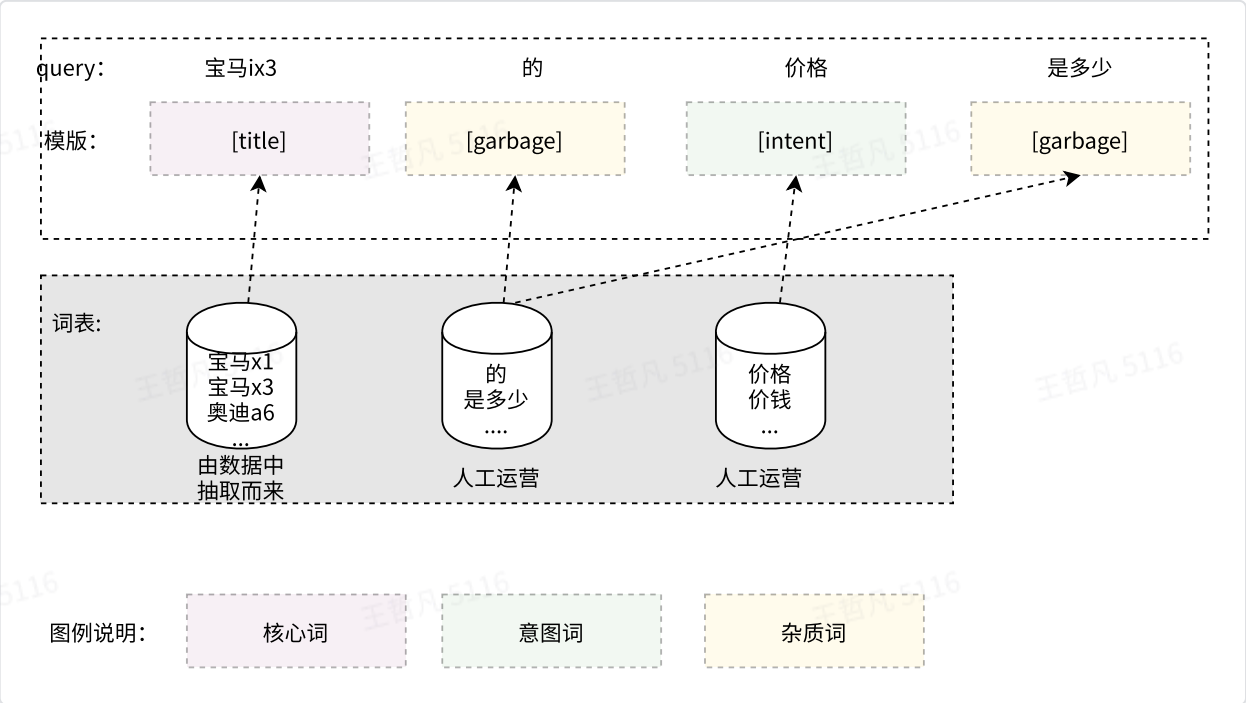
3. 在线检索环节

用户query=“宝马ix3的价格是多少”，分发到特性卡片搜索系统的入口层。

1）意图识别：做初步的流量粗筛。识别出有车型价格消费需求的query。

2）槽位填充

对于该类query，其实可以总结出一套匹配模版，来支持具体的槽位填充。具体过程，可以如下图所示。



用户输入query=“宝马ix3的价格是多少”，会能命中归纳的模版[title][garbage][intent][garbage]。它包含了4个槽位。每个槽位至少包含以下几个维度的信息：

- 槽位属性，比如：核心词；意图词；杂质词等。相关名词介绍
- 数据来源：本地词表（基于字符串命中）

经过本环节，能提取出：title=宝马ix3

在本环节，

- 通过扩展新模版，能覆盖更多的query场景。

📌 比如，模版[title][garbage][title][price_pk]，槽位“price_pk”挂载本地词表（下含内容“谁更好”；“谁更贵”等），就能方便的覆盖 q="宝马x1和宝马x3谁更好"等车型pk等场景的满足

- 通过对模版下词表内容的扩展，能在某个模版/query需求场景下，提升召回覆盖面。

3）检索环节。根据title=奥迪ix3 能在“倒排索引”中提取出 如demo数据中car_id= 53947 以及car_id= 53948的候选结果。这块还可以基于不同的业务需求，引入排序公式去影响最近呈现的结果排序。

4）摘要环节。能在“摘要环节”根据car_id取出对应的摘要结果。

经过上述分析，构建一个配套“可视化的强规则运营平台”的特型卡片搜索系统，能最大程度的在一个企业内，最大化的完成不同工种（技术/pm/运营等）的分工协同，形成很好的规模效应，助力特型卡片的搜索业务快速/敏捷迭代。事实上，国内各大搜索引擎的发展均或早或晚的经历过/经历着这个类似系统的建设和迭代过程。

当然，真实工业生产中，遵从二八原则，类似平台能覆盖一部分场景。但是对于头部/重点的特型卡片类目，其实是有更先进/柔和的召回覆盖方案，有兴趣的话再单独交流。

本次课题，希望同学们能完成一个demo版本的特型卡片搜索系统，也借这个机会，对信息搜索领域有一些初步的认识。

名词说明

- 召回：在搜索引擎中，从候选收录文档中，选出最满足用户query需求的候选文档的过程。
- 资源/垂类，连接数据和搜索需求的满足。以下简称为Resource。
- 召回率：检索出的相关文档数和文档库中所有的相关文档数的比率，衡量的是检索系统的查全率
- 准确率：检索出的相关文档数与检索出的文档总数的比率，衡量的是检索系统的查准率
- 槽位属性：
  - 杂质词：槽位填充过程中，可丢的部分。即使去掉，对query意图的表达也没影响
  - 意图词：综合搜索下，表达一类搜索需求。一般不可省略
  - 核心词：经由结构化数据中抽取聚合而来。一般不可省略

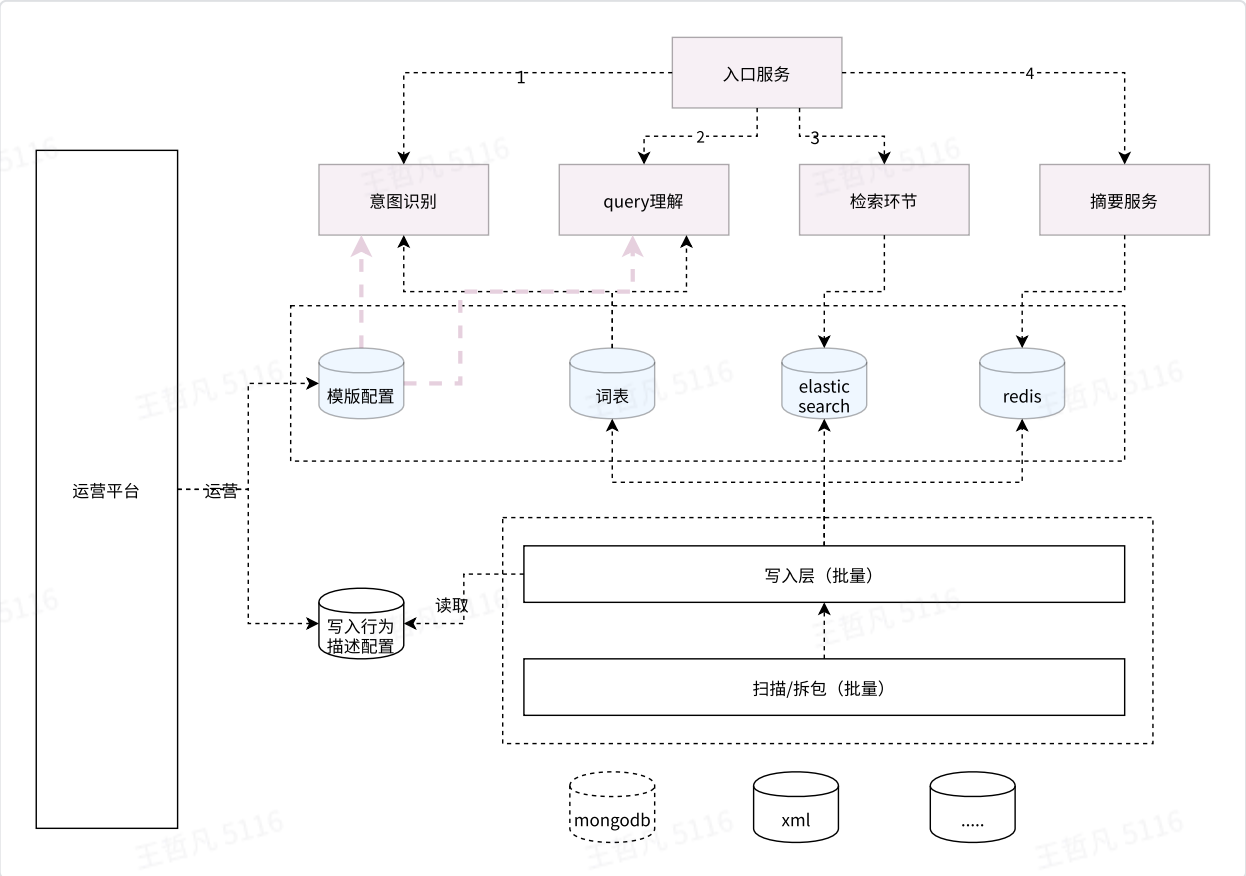
二、项目主要功能及业务流程

基于上一章节的场景分析，本次的特性卡片搜索系统至少包括以下几个部分：

- **搜索服务**：用于在线检索过程，对于给定的询问返回若干卡片摘要。具体又可以划分为以下几个环节：
  - **意图识别**：识别可能潜在的特型卡片意图。
  - **query理解**：匹配各卡片的模板，完成槽位填充。
  - **检索环节**：根据槽位填充结果查询倒排引擎得到数据id等。
  - **摘要服务**：根据id查询redis得到最终的若干卡片摘要结果。
- **运营平台**：用于配置原始结构化数据写入各数据层的“写入行为描述”，以及query理解中用于匹配的模板。
- **结果展示**：用于展示搜索结果页的demo。

除此之外，还需要词表、倒排引擎（elastic search）、redis等**数据层**的存在，用以串联起各个部分与环节。

具体关系可参阅下图：



数据层

在 案例分析，我们提到需要用到倒排索引 + 词表 + kv存储的需求。对相关的数据交互希望遵循以下几个原则：

1、倒排引擎建议直接用elasticsearch（开源；api足以满足需求）

2、kv存储建议选用redis。

3、读写方式 & schema的约定足够通用。能做到整个系统搭建完毕后，新增特型卡片，在本环节，不需要显式的手动的做任何额外的存储/资源的申请

离线建库

流量的在线阶段都是在进行“读”操作：查词表、查ES、查Redis。这些数据的写入肯定是和在线请求异步的，我们称为“离线建库”。

离线服务的意义是把最原始的结构化数据转化成搜索在线阶段可直接读取使用的库（即词表、ES和Redis）。具体来说，可以拆成两个环节：

1、扫描/文件读写&拆包（拆成1条条结构化数据）

> 📌 目前工业界普遍使用xml格式的数据，因此兼容XML类型的数据为项目基础需求
>
> 【可选】支持json格式的数据。
>
> 【可选】有余力又可以了解和使用下mongodb等半结构化数据库。

注意，这块考虑到xml数据是全量版本。因此。本环节拆包后，下发写入环节时，单条目的数据可以是update/insert以及delete行为。

技术上，可以重点考虑下delete功能如何完成

【简单思路】每次更新时，遍历全量xml数据文件得到前后两个版本的数据的diff，并对单条数据做出update/insert/delete的决策

2、完成写入操作。这块依赖“写入行为描述”，完成具体的结构化数据抽取，以及转换/归一化，最终写到不同的数据层里。

```Shell
input:
xml/json文件，内部是原始的结构化数据。
output:
- Redis，可根据卡id+docid查询到对应数据
- ES，可根据ESquery查询到对应docid
- 词表，可根据卡id+字段名查询到所有词
```

这块的建库行为如何触发呢?

- 【必须】本系统接受通过crontab形式做例行建库
- 【必须】联动 运营平台 实现 人工触发建库行为 的功能

在线检索服务

整体检索服务的对外交付，建议是http server的形式。

系统构建时，可以简化处理，把所有子服务归拢到一个大的服务中来。但是也期望【可选】可以遵从微服务的思想构建。服务之间使用rpc方式调用

接口约定：

- Input: query
- output：符合query需求的若干个资源的结果内容列表。每个元素下含如下内容：
  - Resource（类型名）
  - 符合query需求的 特定特性下的 结构化数据 列表

Demo output可以是如下：

JSON

```
[
    {
        "Resource": "A",
        "datas": [
            {
                //raw data here
            }
        ]
    },
    {
        "Resource": "B",
        "datas": [
            {
              //raw data here
            },
            {
              //raw data here
            }
        ]
    }
]
```

意图识别（也可以考虑微服务/独立服务）

本服务的意义是产出本次请求的意图（要对哪些卡片进行召回）。当然，对于简单的请求量比较小的搜索架构，这一步骤是不必要的，全量请求进行检索阶段即可。但是在真实场景中，如果流量大到一定的程度，全部请求调用全部链路是一种奢侈，以头条的真实场景车系卡为例，与此相关的搜索流量连1%都不到，全部流量都进行车系卡的检索无疑是不现实的，所以这一步骤的最重要意义是过滤无关流量。另外，意图的精准程度也影响着用户体验，q=“母亲节图片”和q=“母亲节”的文本差别

很小，但是用户意图差别很大。要求越高，识别难度也越大，真实场景中往往采用工程和策略算法相结合的方式，本课题不要求较高的精准程度。

在精准程度不高的前提下，我们约定：query中包含特型卡A的核心文本就算作有特型卡的意图。比如query中有“宝马”，而“宝马”在原始数据的“brand_name”字段中，同时“brand_name”字段是车系卡的核心字段（可在写入行为描述中指定），那我们就可以认为query有车系特型卡的意图。

【简单思路】关于实现方式，一个简单可行的方式是AC自动机，或者Trie树。从根节点到叶子节点的路径串成一个文本，每个文本都有多个意图，如果query的某个substring能够通过Tire树的匹配，就能顺利产出其包含的意图。

```
Input: query
output：数组形式结果。每个元素下含
- 特型卡name/id
```

query理解（也可以考虑微服务/独立服务）

本环节用来完成（若干）特定特性卡片下的槽位填充工作。

Input: query + intent_list。（intent_list为Resource数组）

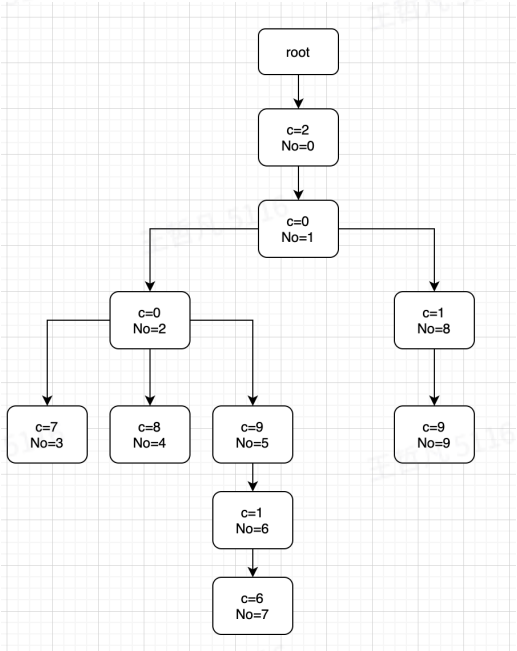
output：query + Resource纬度数组（下含Resource名称 + query在本Resource下命中到的模版 & 更具体的槽位信息）

【简单思路】这里提供query模板匹配的一种简单思路。

以单个词槽的匹配为例，离线产出了一份 `xxx_date` 词典，如下：

```
xxx_date
2007
2008
2009
200916
2019
```

根据词典，可产出如下Trie树：



如此一来，将“abcd是否匹配`xxx_date`的问题”转为“abcd是否能完整遍历树的单个分支”的问题。

同理，多个词槽的匹配也可以用此思路。

【可选】为了提高召回覆盖，这块除了做 案例分析提到的字符匹配外，也期望能支持同义词词表载入能力。

demo：q=帝都天气怎么样； 天气类目下，只有“北京”的数据。正常情况下，是无法找回的。但是借助同义词词表：

```
Plain Text
1  帝都:北京 //这块技术实现上，完成 帝都 到 北京 的建边关系
```

至于同义词词表如何生成，一般是人工运营 + 离线知识图谱/实体挂接/挖掘等技术完成。本demo项目中，在配套运营平台提供运营入口即可。

检索环节（也可以考虑微服务/独立服务）

Input: query分析结果

output：docId（keyword）

本服务的功能是根据query分析结果，检索得到符合条件的docId。

举例：

query=北京宝马二手车，分析结果：`【city:北京】（核心词）【brand:宝马】（核心词）【intent:二手车】（意图词）`

使用query分析结果中的核心词，构建查询语句查询ES。

这块的查询语句设计需要结合“数据层”和离线建库中的描述，设计出足够通用的查询语句。

摘要（也可以考虑微服务/独立服务）

对用户来说，最终数据是呈现到前端页面的一条条结果，每条结果的展现内容我们称为“摘要”。

摘要服务的意义就顾名思义，根据数据的唯一键（卡id+docid）获取数据的展现内容。

实现方式上，通过Key取Value，真实场景最常见的就是Redis。

Shell

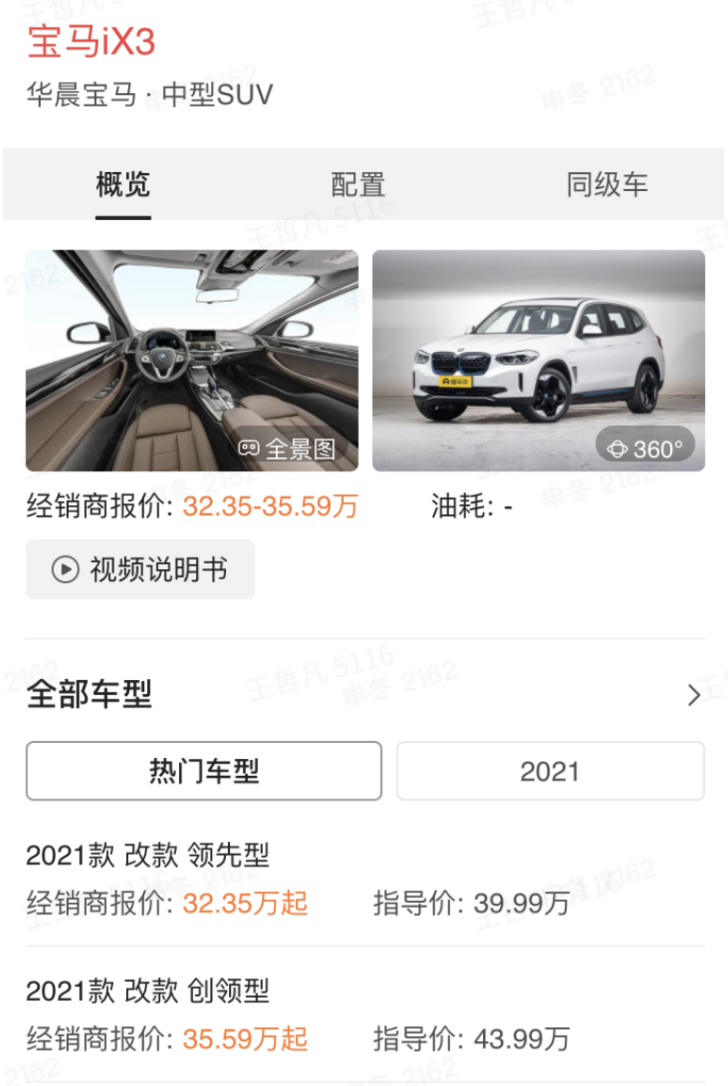
```
Input: 数组形式,每个元素包含:
- 特型卡id/name
- docId

output: 数组形式，每个元素包含:
- 特型卡id/name
- docId
- 本条数据的摘要

```

结果渲染

【在线检索服务】返回的结构化数据检索结果后，可以在此处做结果展示（只做静态页面渲染即可）



必备实现的渲染要素包括：

- 标题（文本）
- 列表结果（文本）
- 图片渲染/展示
- 【可选】query关键字飘红展示
- 【可选】动态页面效果、用户输入等交互、嵌入视频等

运营平台

这个平台主要是两个功能：

1. 【必须】产出“模版配置” & “写入行为描述”，供在线/离线服务使用
2. 【必须】针对上述配置，提供和实现可视化/友好的交互页面，降低配置运营成本。
3. 【必须】提供人工触发全量离线建库的功能模式
4. 【必须】提供各种内外部运营人员的登录与权限系统（权限设置分配合理即可）
5. 【必须】提供直接编辑离线数据的功能模式和交互页面（线上的搜索引擎经常有快速修改页面展示的需求，比如新闻类特型卡，此时调用全量建库是生效较慢的，直接编辑库中的数据才是最快的方式。注意，与全量扫描离线建库的产物可能存在版本冲突，合理解决即可）

6. 【可选】提供同义词词表的编辑

模板配置

数据说明：

(Resource代指特型卡id)

JSON

```json
{
  "Resource":"demo_resource",
  "rule_recall_setting_list": [
    {
      "garbage_dict_list": [
        "parser_墨迹天气_200_garbage"
      ],
      "pattern_list": [
        {
          "pattern_item_array": [
            [
              {
                "data_pointer": "name",
                "data_source_type": 0,  //数据来源，0：从离线结构化数据中抽词而来；4:本地运营的自定义本地词表
                "field_name": "name",
                "query_item_type": 0. //描述槽位属性。0:核心词；1:杂质词；2:意图词
              }
            ],
            [
              {
                "data_pointer": "parser_墨迹天气_200_intent",  //在线服务 和 运营平台 提前 做好这个词表的相对路径约定
                "data_source_type": 4,
                "field_name": "intent",
                "query_item_type": 2
              }
            ]
          ],

          "use_common_garbage_dict": true,
          "use_synonym_dict": true,  //可选功能
        }
      ],
      "pre_processors": [

      ],
      "synonym_dict_list": [
        "parser_synonym_墨迹天气_200_1622687488" //同义词词表
      ]
    }
  ]
}
```

参照的页面设计：

写入行为描述

可以参照的数据协议约定：

JSON

```
{
  "Resource":"demo_resource",
  "write_setting": [
    "a.b.c":{  //指定原始结构化数据中的字段路径
            "dump_digest":"true",  //本字段是否需要dump摘要
            "dump_invert_idx":"true", //本字段是否需要dump倒排
            "dump_dict":"true", //本字段是否需要dump词表
    },
    "b.g":{
            "dump_digest":"true",
            "dump_invert_idx":"false",
            "dump_dict":"true",
    }

  ]
}
```

对应的一种JSON格式原始数据（XML也类似）：

JSON

```
{
    "a": {
        "b": {
            "a": XXX,
            "c": XXX //对应write_setting中的第一条
        }
    },
    "b": {
        "g": XXX //对应write_setting中的第二条
    }
}
```

三、需求说明

必备功能

1、章节2中，非显式提到“可选”的均为期望必须实现的基本功能

2、结果渲染，需要对基本的前端技术有所使用和熟悉

可选功能

1、运营平台产出配置/离线抽词环节产出的本地词表，如何和在线/离线服务做更及时/自动化的同步和生效

2、其他在章节2提到的可选功能

四、项目联系人

联系人：申冬

微信：

邮箱：shendong.wonder@bytedance.com

五、项目QA

Q：原始的结构化数据从哪里来？

A：后续经过字节跳动内部数据申请后提供。

Q：为什么需要支持的结构化数据类型为XML格式？

A：XML格式为目前工业界常用格式，这种设计也是为了与业界对齐。

Q：“写入行为描述”部分的约定是否是确定的？

A：给出的仅为一种较标准的示例，具体软件设计中由同学们自行设计即可。