1. 新闻资讯项目

1.1数据迁移

数据迁移主要分为两部分：

1. 自有新闻数据迁移
   1. 相关数据库：database -> cms，table ->article, cms\_tmp
   2. 相关代码：MarineOnline\_Service -> data\_migration -> src -> migrate\_cms.py
2. 爬虫爬取的第三方新闻数据迁移
   1. 相关数据库：database -> mol，table -> news, t\_crew\_data, t\_crew\_data\_file\_rel, t\_crew\_file, news\_source\_config
   2. 相关代码：MarineOnline\_Service -> data\_migration -> src -> migrate\_crawl.py, migrate\_picture.py

辅助部分：

MarineOnline\_Service -> data\_migration -> src -> settings.py: 设置配置文件

MarineOnline\_Service -> data\_migration -> src -> schedule.py: 设置任务定时器

相关数据库表：

1）cms.article：自有新闻内容

2）cms.cms\_tmp：自有新闻迁移中间表

3）mol.t\_crew\_data：第三方爬虫新闻内容表

4）mol.news：news内容正式表

5）mol.news\_picture：新闻图片信息表

6）mol.news\_tag：新闻关键词表

1.2 自有新闻迁移流程 migrate\_cms.py

1）pre\_processing：将cms.cms\_article表中对应字段插入到迁移中间表格 cms.cms\_tmp中，并根据业务需求将相关字段进行处理。（细节见迁移规则）

2）migrate\_news：将上一步预处理完毕的内容，从迁移中间表格cms.cms\_tmp插入到正式表mol.news中。

3）migrate\_picture：将自有新闻的图片信息从cms.cms\_tmp表格中插入到mol.news\_picture

4）migrate\_tag：将自有新闻的tag从cms.cms\_tmp中插入到mol.news\_tag表。

1.3 第三方爬虫新闻迁移流程 migrate\_crawl.py

1）set\_level：根据业务规则将第三方新闻的图片初步分类为L1和L3，细节见迁移规则。

2）set\_template：根据业务规则将第三方新闻的图片template字段设置为L\R\T\S，细节见迁移规则。

3）set\_content\_text：从正文内容取前800个字符作为新闻摘要。

4）migrate\_crawl：将相应字段从mol.t\_crew\_data中迁移到正式表mol.news

1.4 图片迁移流程 migrate\_picture.py

1）load\_data：从mol数据库中获取图片信息相关字段，并转换成pandas dataframe

2）set\_order：根据业务规则，给每条新闻对应的图片设置图片顺序

3）write\_to\_mysql：将处理好的data frame对应内容插入到mol.news\_picture中

4）update\_pic\_flag：将pic\_flag设置为1表示图片已经完成迁移。

1.5迁移规则

1表格关联规则

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| cms.article | | cms.cms\_tmp | | mol.news | mol.t\_crew\_data | | mol.t\_crew\_data\_file\_rel | | mol.t\_crew\_file | |
| id | → | id | → | id | ← | id |  | id |  | id |
|  |  |  |  |  |  | url\_crc | ← | data\_id |  |  |
|  |  |  |  |  |  |  |  | file\_url\_crc | ← | file\_url\_crc |

2. 数据迁移频率

1）CMS 到 mol.news主表：距上次执行完1分钟后执行一次。

2）第三方爬虫库到 mol.news 主表：距上次执行完2小时后执行一次

3 图片模板设置规则（仅针对第三方爬虫库数据）：

1. UPDATE mol.t\_crew\_file SET `level` = 'L1' WHERE width >= 879 AND height >= 261 AND width/height > 2 AND `level` IS NULL  
   说明：在mol.t\_crew\_file表中，图片size符合 width >= 879 并且 height >=261并且width：height>2的数据，将其临时字段 `level` 设置为 L1；
2. UPDATE mol.t\_crew\_file SET `level` = 'L3' WHERE width >= 195 AND height >= 146 AND width/height <= 2 AND `level` IS NULL  
   说明：在mol.t\_crew\_file表中，图片size符合 width >= 195 并且 height >=146并且width：height<=2的数据，将其临时字段 `level` 设置为 L3；
3. 针对同一条新闻的图片数据，如果其 `level`字段为L3 并且图片的数量大于等于1张，那么在mol.t\_crew\_data中将template 字段暂时随机设置为 1或者2，表示一张小图片左侧或者右侧；
4. 针对同一条新闻的图片数据，如果其 `level`字段为L3 并且 width >= 275 并且 height >= 205 并且 width:height <= 2 并且图片的数量大于等于3张，那么在mol.t\_crew\_data中将template 字段设置为 T，表示三张小图片；
5. 针对同一条新闻的图片数据，如果其 `level`字段为L1 并且图片的数量大于等于1张，那么在mol.t\_crew\_data中将template 字段设置为 S，表示一张大图片；
6. 针对步骤3中，将mol.t\_crew\_data表中template字段为1的数据更新为L，表示左侧一张图片，将template字段为2的数据更新为R，表示右侧一张图片；
7. 上述1-6步全部完成后，将mol.t\_crew\_data中字段template 为非空的数据其tem\_flag字段由0设置为1，表示该数据存在符合规则的图片且模板字段已经设置完毕，否则其 template为空且 tem\_flag为默认值0.

4. 第三方爬虫库数据迁移到mol.news主表

1. UPDATE mol.t\_crew\_data  
   SET content\_text = SUBSTRING(content FROM 1 FOR 800)  
   WHERE content\_text IS NULL AND template IS NOT NULL  
   说明：针对经过图片筛选之后的数据（即mol.t\_crew\_data表中template字段不为空），设置 mol.t\_crew\_data中content\_text字段为content字段前800个字符。
2. 将mol.t\_crew\_data中对应数据整体迁移到mol.news表中，字段对应规则如下：

|  |  |  |
| --- | --- | --- |
| mol.news |  | mol.t\_crew\_data |
| id | ← | id |
| template | ← | template |
| title | ← | title |
| content\_text | ← | content\_text |
| source\_code | ← | resource |
| source | ← | source\_name |
| source\_url | ← | url |
| external\_post\_time | ← | resource\_publish\_time |

1. 步骤2中迁移限制说明：news二期新增了网站开关功能，迁移时必须满足template字段不为空 并且 mol.news\_source\_config表中对应网站的字段source\_status=1（即开关为开启状态）才能迁移。

5. 第三方爬虫图片迁移到mol.news\_picture

1. 针对mol.t\_crew\_data表中template字段不为空并且 mol.t\_crew\_file表中oss\_url 不为空的图片数据，进行 order 设置；
2. 在mol.news\_picture中写入图片数据，news\_id 对应 mol.t\_crew\_data中的id，url对应mol.t\_crew\_file表中oss\_url，order对应步骤1中设置的order；
3. 图片迁移完成后，将mol.t\_crew\_data中的pic\_flag由默认值0设置为1，表示已经迁移图片。

6. 第三方爬虫数据关键字生成过程

1. 从mol.t\_crew\_data中提取模型用数据（限制条件：pic\_flag =1 即存在满足规则的图片并已经完成迁移），调用模型接口针对每篇新闻生成5个关键字；
2. 将新闻id和对应的关键字依次写入到mol.news\_tag表中news\_id, tag字段；
3. 上述步骤完成后，将mol.t\_crew\_data表中keywords\_flag字段由默认值0设置为1，表示对应新闻已完成关键字生成并存入到mol.news\_tag表中。

7. CMS库迁移到mol.news主表

1. 将cms.article表中部分对应字段迁移到中间表cms.cms\_tmp中，限制条件为：cms.article表中字段is\_active = 1并且 cms.article 表中 code 字段在（article, market, people）中，对应规则如下：

|  |  |  |
| --- | --- | --- |
| cms.cms\_tmp |  | cms.article |
| id | ← | id |
| tag | ← | code |
| title | ← | subject |
| resource\_publish\_time | ← | update\_data |

1. 将cms.cms\_tmp表中迁移过来的数据source\_name 字段设置为”Marine Online”，template字段统一设置为L，tem\_flag字段由默认值0设置为1，created\_by字段设置为cms\_migrator增加唯一标识
2. 根据cms新闻数据的id和配置的url前缀拼接成完整的url写入到cms.cms\_tmp表中url字段；
3. 调用自定义函数 json\_extract\_c 从cms.article表data字段中提取图片链接写入到cms.cms\_tmp表pic\_url字段中；
4. 根据cms.category\_match表中category字段给cms.cms\_tmp表中category赋值；
5. 调用自定义函数 json\_extract\_c 从cms.article表data字段中提取正文内容以纯文本的形式存放到 cms.cms\_tmp字段中。
6. 将cms.cms\_tmp表中数据整体迁移到mol.news表中，字段对应规则如下：

|  |  |  |
| --- | --- | --- |
| mol.news |  | cms.cms\_tmp |
| id | ← | id |
| template | ← | template |
| title | ← | title |
| content | ← | content |
| source | ← | source\_name |
| source\_url | ← | url |
| external\_post\_time | ← | resource\_publish\_time |
| create\_by | ← | created\_by |

1. 根据二期新增内容，将mol.news表中来源于CMS的新闻数据source\_code字段设置为marineonline，并且将mol.news表中非人工创建的CMS新闻数据is\_original字段由默认值0设置为1（即限制条件为： source\_code =’marineonline’ AND create\_by = ‘cms\_migrator）；
2. 上述过程完成后，后端会从mol.news表中content字段提提取出正文内容存放到content\_text字段中，并更新status值为PD。

8. CMS图片数据迁移到mol.news\_picture表

1. 将cms.cms\_tmp表中id, pic\_url 字段对应迁移到mol.news\_picture表中news\_id,url字段；
2. 步骤1完成后将cms.cms\_tmp表中pic\_flag字段由默认值0设置为1，表示图片已经迁移完成；
3. 将mol.new\_picture表中来源于cms的数据 order字段设置为1。

9. CMS关键字迁移到mol.news\_tag表

1. 将cms.cms\_tmp表中id, tag字段对应写入到 mol.news\_tag表中news\_id, tag字段；

将cms.cms\_tmp表中tag\_flag字段由默认值0设置为1，表示tag迁移完成。

1. 关键词提取项目

2.1 相关代码和数据库

主要部分：

1）MarineOnline\_MachineLearning -> keywords -> src -> genetate\_total\_corpus.py: 生成语料库

2）MarineOnline\_MachineLearning -> keywords -> src -> extract\_en.py: 提取关键词

辅助部分：

1）MarineOnline\_MachineLearning -> keywords -> src -> settings.py: 设置配置文件

2）MarineOnline\_MachineLearning -> keywords -> src -> schedule.py: 设置任务定时器

3）MarineOnline\_MachineLearning -> keywords -> src -> stopwords\_english.txt: 英文停词表

相关数据库表格

1. 爬虫表：mol.t\_crew\_data
2. 关键词表：mol.news\_tag

2.2 语料库生成流程 generate\_total\_corpus\_en.py

1）load\_data: 从数据库读取符合规则的全部新闻内容，包括id, title, content，返回值类型为pandas dataframe

2）load\_stopwords：从英文停用词表中读取停用词，返回值类型为 list。

3）pre\_processing：参数：全部新闻正文内容的news\_list。循环读取list中的每一篇新闻正文内容，依次进行： a）正则化提取英文字母，其他字符使用空格替代；b）分词并去掉重复的空格；c）词干提取，词型还原；d）根据停用词表去掉停用词；e）所有字母转换为小写。处理完毕后append到corpus列表，返回值类型为list。

4）save\_corpus：将第三步处理好的新闻正文内容保存到本地文本文件。

2.3 提取关键词流程 extract\_en.py

1）load\_corpus：从本地文本文件中读取2.2中处理好的语料库，返回值类型为list。

2）load\_data：从数据库中读取尚未提取关键词的新闻，获取对应的id、title、content字段，返回值类型为pandas dataframe。

3）load\_stopwords：从英文停用词表中读取停用词，返回值类型为 list。

4）针对每条待提取关键词的新闻通过pre\_processing函数进行预处理，包括：a）正则化提取英文字母，其他字符使用空格替代；b）分词并去掉重复的空格；c）词干提取，词型还原；d）根据停用词表去掉停用词；e）所有字母转换为小写。返回值类型为list。

5）TF-IDF算法：相关函数tf()、 document\_frequency()、 idf()

6）针对步骤4预处理好的每条新闻，调用get\_keywords()函数提取出每篇新闻的top5关键词，返回值类型为list。

7）write\_to\_mysql：将每篇新闻提取的结果写入到数据库 news\_tag表中

8）update\_keywords\_flag：将爬虫表中keywords\_flag由0设置为1，表示该新闻已经提取关键词。

1. 新闻自动分类项目（尚未上线）

3.1 相关代码和数据库

主要部分：

1）MarineOnline\_MachineLearning -> news\_classify\_en -> src -> pre\_processing.py: 将数据库提取的新闻数据和类别标签进行预处理

2）MarineOnline\_MachineLearning -> news\_classify\_en-> src -> pre\_model\_data.py: 生成语料库

3）MarineOnline\_MachineLearning -> news\_classify\_en-> src -> build\_model.py: 生成语料库

4）MarineOnline\_MachineLearning -> news\_classify\_en -> src -> prediction.py: 提取关键词

辅助部分：

1）MarineOnline\_MachineLearning -> news\_classify\_en -> src -> settings.py: 设置配置文件

数据库

1）mol.news

3.2 预处理过程 pre\_processing.py

1）laoda\_data：从mol.news数据库中获取已经带有分类标签的新闻，返回值类型为list

2）clean\_str：辅助函数，用于清洗和标准化字符串

3）pre\_processing：针对每篇新闻的title调用clean\_str函数进行清洗和标准化，同时把标签类别进行数字化，最后把处理好的新闻语料库保存为本地文本文件。

3.3 模型数据准备过程 pre\_model\_data.py

1）create\_word\_dict：针对当前提取的所有带类别标签的新闻标题进行数字化转换，并将结果以字典的形式保存到本地文件，一个字符对应一个数字编码。

2）create\_data\_list：将3.2中预处理好的新闻语料库拆分成训练集和测试集，每10条新闻取一条数据为测试集，其他为训练集，同时使用word字典将训练集和测试集数字化。

3）将生成的训练集和测试集保存到本地文件备用。

3.4 生成模型过程 build\_model.py

1）train\_extractor: 读取训练集数据，并进行随机洗牌，存入生成器，返回值类型为paddle reader，并调用data\_mapper将特征和标签进行配对。

2）test\_extractor：读取测试集数据，并进行随机洗牌，存入生成器，返回值类型为paddle reader，并调用data\_mapper将特征和标签进行配对。

3）CNN\_net：构造CNN神经网络并设置相应参数。

4）通过fluid.layers.data定义输入数据words和label，并初始化CNN神经网络；fluid.layer.cross\_entropy设置损失函数；fluid.optimizer设置优化方法；fluid.Executor设置CPU或者GPU执行器，fluid.DataFeeder设置数据映射器

5）设置总训练轮数为10轮，通过train\_reader和test\_reader将数据未入训练器中进行训练，训练完毕后将模型数据保存到本地文件，同时训练过程中输出模型的准确性。

3.5 预测过程 prediction.py

1）get\_dict\_code：针对待预测的新闻内容初始化单词字典

2）初始化模型预测器，调取3.4中保存好的模型数据，对待预测新闻进行预测并输出结果。

1. 港口匹配项目

4.1 相关代码和数据库

主要部分：

1）MarineOnline\_MachineLearning -> port\_match-> src -> portMatch2.py: 港口匹配升级版。

辅助部分：

1. MarineOnline\_MachineLearning -> port\_match-> src -> portMatch.py: 港口匹配初版（已停止使用）。
2. MarineOnline\_MachineLearning -> port\_match-> src -> MatchTest.py: 港口匹配手动测试辅助工具。
3. MarineOnline\_MachineLearning -> port\_match-> src -> MatchTool.py: 港口匹配批量测试辅助工具。
4. MarineOnline\_MachineLearning -> port\_match -> src -> settings.py: 设置配置文件。
5. MarineOnline\_MachineLearning -> port\_match -> src -> schedule.py: 设置任务定时器。

相关数据库表格

1. ports.port\_dict表：存放港口库数据，来源为：a）标准库表ports\_port，source=0；b)编辑距离计算得出，source=1；c）人工手动添加，source=3。可信度字段 flag，默认值为0，来源于标准库对应flag为0，编辑距离计算得到的和人工手动添加的港口名称数据的flag默认为0，经人工复核确认无误将flag设置为1。
2. vessels.ais\_info表：需要更新匹配的主表。
3. vessels.ais\_track表：用于更新历史数据。

4.2 港口匹配过程 portMatch2.py

1）get\_port\_code\_name：从数据库ports.port\_dict中读取port\_code和port\_name字段，并将其配对转换为字典类型返回。

2）get\_destination：从数据库vessels.ais\_info中读取需要进行匹配的数据，字段包括 mmsi, imo, eta, match\_end\_port, match\_end\_type, score\_end，返回值类型为list。

3）port\_match：循环读取上一步的每个destination根据业务规则进行港口的模糊匹配，详见匹配规则。

4）通过multiprocessing.pool进行多进程匹配。core\_num设置进程数，pool.apply\_async调用port\_match进行匹配和更新数据库。

4.3 匹配规则

1. 判断历史匹配结果是否为空值，如果为空，则将history\_name和history\_type设置为’’, history\_score设置为0.
2. 调用辅助函数filter\_des将用户输入的destination值进行初步过滤，包括去掉用户输入值中的字符串’OFF’，如果存在‘>’则将输入值拆分成前后两部分des\_start和des\_end，如果不存在‘>’，则将输入值设置为des\_end，des\_start设置为空。
3. 当用户的输入值仅存在des\_end的时候（即不存在“>”, des\_start为空），先判断初步过滤后的des\_end字符串长度是否小于等于3或者存在特定的字符串比如“order”,“arned”，“guard”，如果满足该条件，则将end\_name和end\_type设置为“Unknown”，end\_score设置为0，当本次的des\_end值和历史值不同时，同步更新 vessels.ais\_info和vessels.ais\_track表，若和历史值相同，则仅更新匹配时间match\_time。如果不满足字符串长度小于等于3的条件，则首先在现有港口字典中进行查找，如果存在该key对应的value，则直接返回结果，并将end\_type 设置为“dictionary”，“end\_score”设置为100，同时根据相应规则更新vessels.ais\_info和vessels.track表。如果现有港口字典中不存在该key，则调用编辑距离算法进行计算并排列，取top1值返回，将end\_type设置为“editDistance”，end\_score取计算分数，根据规则更新vessels.ais\_info和vessels.track表的同时，将当前计算出来的key和value以字典的方式新增并写入到ports.port\_dict表中。
4. 如果用户输入的值包含des\_start和des\_end两部分，则分别针对这两部分按照步骤3）进行匹配取值，最后同时更新des\_start和des\_end字段到vessels.ais\_info表。