Apache Flink

Alink: 提升基于Flink的机器学习平台易用性

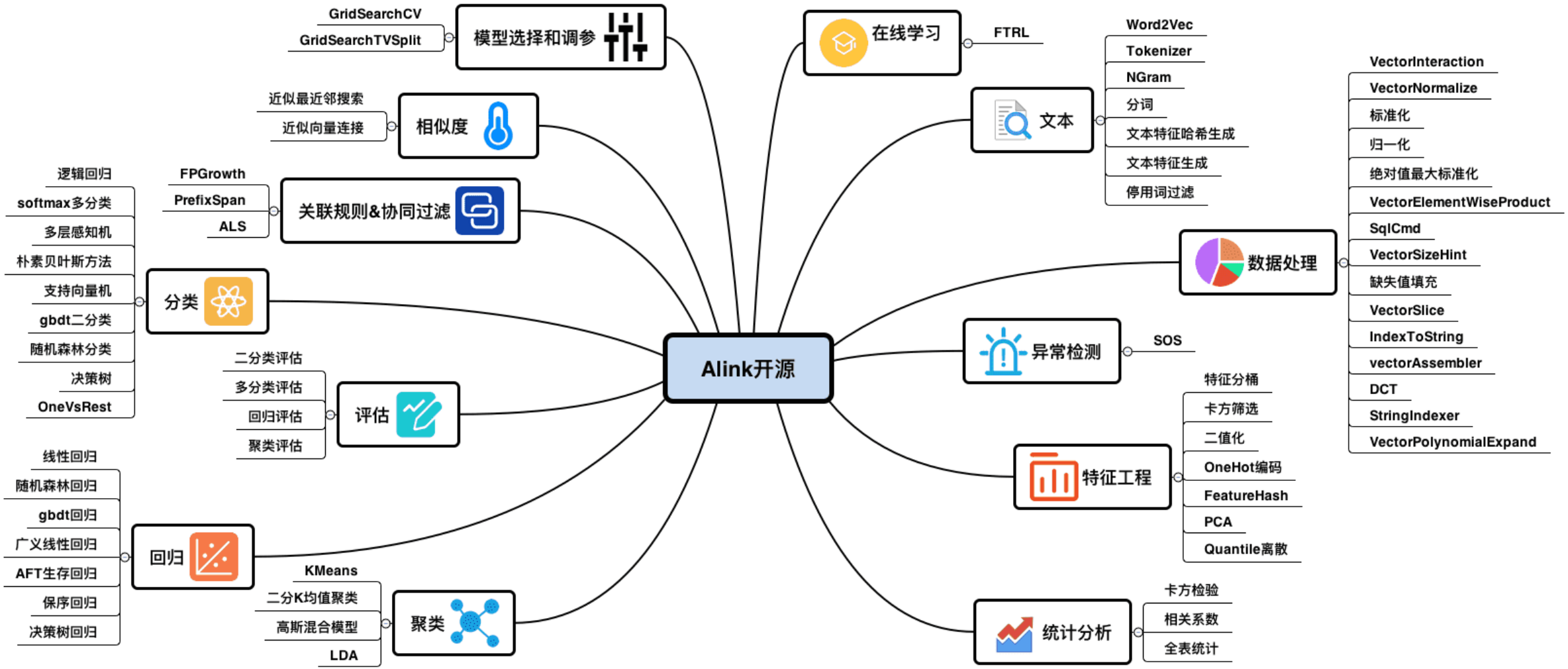
杨旭 · 阿里巴巴计算平台事业部 / 资深算法专家

北京 – 2020年4月25日

Alink进展总览

- Alink version 1.0
  - 2019年11月在Flink Forword Asia大会上宣布开源
- Alink version 1.0.1
  - 2019年12月发布，解决一些场景下PyAlink的安装问题
- Alink version 1.1.0
  - 2020年02月发布，支持Flink 1.10和Flink 1.9，PyAlink兼容PyFlink
  - 支持发布到Maven中央仓库和PyPI
- Alink version 1.1.1
  - 2019年04月发布，提升使用体验，在参数检查方面更加智能

Alink version 1.0



Alink version 1.1.0

- 支持Flink 1.10和Flink 1.9
- 与PyFlink兼容
- 发布到Maven中央仓库和PyPI
- 改进UDF / UDTF功能
- 支持多版本的Kafka数据源

使用Maven构建Alink项目

- 第一步，创建maven项目
- 第二步，修改pom文件，导入Alink相关jar包
  - Flink-1.10 的 Maven 依赖
  - Flink-1.9 的 Maven 依赖
- 第三步，拷贝修改Alink Java示例代码
  - KMeansExample.java
- 第四步，构建运行
- 详细过程
  - https://zhuanlan.zhihu.com/p/110059114

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>AlinkMavenExample</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>com.alibaba.alink</groupId>
            <artifactId>alink_core_flink-1.10_2.11</artifactId>
            <version>1.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-streaming-scala_2.11</artifactId>
            <version>1.10.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-table-planner_2.11</artifactId>
            <version>1.10.0</version>
        </dependency>
    </dependencies>

</project>
```

PyAlink 1.1.0 安装实践

一、安装PyAlink之前需要准环节，在不同操作系统上有差异。

- PyAlink安装准备——MacOS https://zhuanlan.zhihu.com/p/110898678
- PyAlink安装准备——Windows https://zhuanlan.zhihu.com/p/97020481
- PyAlink安装准备——阿里云服务器 https://zhuanlan.zhihu.com/p/110898735

二、PyAlink从1.1.0版本开始发布到PyPI，安装操作更加方便。

- 如何安装最新版本PyAlink？https://zhuanlan.zhihu.com/p/110944464

三、对于之前安装过PyAlink的用户，在安装前需要先进行卸载。

- PyAlink的版本查询、卸载旧版本 https://zhuanlan.zhihu.com/p/109949429

PyAlink批式任务在notebook上运行

```
from pyalink.alink import *

## 一个 Batch 作业的例子
useLocalEnv(2)
## prepare data
import numpy as np
import pandas as pd
data = np.array([
    [0, 0.0, 0.0, 0.0],
    [1, 0.1, 0.1, 0.1],
    [2, 0.2, 0.2, 0.2],
    [3, 9, 9, 9],
    [4, 9.1, 9.1, 9.1],
    [5, 9.2, 9.2, 9.2]
])
df = pd.DataFrame({"id": data[:, 0], "f0": data[:, 1], "f1": data[:, 2],
"f2": data[:, 3]})
inOp = BatchOperator.fromDataframe(df, schemaStr='id double, f0 double, f1
double, f2 double')
FEATURE_COLS = ["f0", "f1", "f2"]
VECTOR_COL = "vec"
PRED_COL = "pred"

vectorAssembler = (
    VectorAssembler()
    .setSelectedCols(FEATURE_COLS)
    .setOutputCol(VECTOR_COL)
)
kMeans = (
    KMeans()
    .setVectorCol(VECTOR_COL)
    .setK(2)
    .setPredictionCol(PRED_COL)
)
pipeline = Pipeline().add(vectorAssembler).add(kMeans)
pipeline.fit(inOp).transform(inOp).firstN(9).collectToDataframe()
```

```
from pyalink.alink import *

## 一个 Batch 作业的例子
useRemoteEnv("10.101.**.**", 31805, 2)
## prepare data
import numpy as np
import pandas as pd
data = np.array([
    [0, 0.0, 0.0, 0.0],
    [1, 0.1, 0.1, 0.1],
    [2, 0.2, 0.2, 0.2],
    [3, 9, 9, 9],
    [4, 9.1, 9.1, 9.1],
    [5, 9.2, 9.2, 9.2]
])
df = pd.DataFrame({"id": data[:, 0], "f0": data[:, 1], "f1": data[:, 2], "f2": data[:,
3]})
inOp = BatchOperator.fromDataframe(df, schemaStr='id double, f0 double, f1 double, f2
double')
FEATURE_COLS = ["f0", "f1", "f2"]
VECTOR_COL = "vec"
PRED_COL = "pred"

vectorAssembler = (
    VectorAssembler()
    .setSelectedCols(FEATURE_COLS)
    .setOutputCol(VECTOR_COL)
)
kMeans = (
    KMeans()
    .setVectorCol(VECTOR_COL)
    .setK(2)
    .setPredictionCol(PRED_COL)
)
pipeline = Pipeline().add(vectorAssembler).add(kMeans)
pipeline.fit(inOp).transform(inOp).firstN(9).collectToDataframe()
```

PyAlink流式任务在notebook上运行

- 本地运行

```
from pyalink.alink import *

## 一个 Stream 作业的例子
## 唯一参数表示并行度
useLocalEnv(2)
source = CsvSourceStreamOp() \
    .setSchemaStr(
    "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string") \
    .setFilePath("http://alink-dataset.cn-hangzhou.oss.aliyun-
inc.com/csv/iris.csv")
source.print()
StreamOperator.execute()
```

- 集群运行

```
from pyalink.alink import *

## 一个 Stream 作业的例子
useRemoteEnv("10.101.**.**", 31805, 2, localIp="30.39.**.**")
source = CsvSourceStreamOp() \
    .setSchemaStr(
    "sepal_length double, sepal_width double, petal_length double, petal_width
double, category string") \
    .setFilePath("http://alink-dataset.cn-hangzhou.oss.aliyun-inc.com/csv/iris.csv")
source.print()
StreamOperator.execute()
```

PyAlink基于PyFlink 进行整合

- 提供了 Alink Operator 与 PyFlink Table 的互相转换，方便串联 Flink 和 Alink 的工作流。

```
###  get_mlenv.py
from pyalink.alink import *
env, btenv, senv, stenv = getMLEnv()
### 使用 PyFlink 接口，与 Table 进行互转
table = stenv.from_elements([(1, 2), (2, 5),
(3, 1)], ['a', 'b'])
source = TableSourceStreamOp(table)
source.print()
StreamOperator.execute()
```

- 新提供了 getMLEnv 接口，能直接使用 flink run –py *.py 往集群提交作业。

直接运行脚本

```
python kmeans.py
```

向集群提交作业

```
PYFLINK_PATH=`python -c "import
pyflink;print(pyflink.__path__[0])"`
${PYFLINK_PATH}/bin/flink run -m 10.101.**.**:31805 -py
kmeans.py -p 4
```

读写Kafka示例

定义kafka数据源operator

```
data = KafkaSourceStreamOp()\
        .setBootstrapServers("localhost:9092")\
        .setTopic("iris")\
        .setStartupMode("EARLIEST")\
        .setGroupId("alink_group")
```

使用json提取组件解析kafka中的数据、数据类型转换

```
json_parser = JsonValueStreamOp()
        .setSelectedCol("message")
        .setOutputCols(["sepal_length", "sepal_width", "petal_length",
"petal_width", "category"])
        .setJsonPath(["$.sepal_length", "$.sepal_width", "$.petal_length",
"$.petal_width", "$.category"])

data = data.link(json_parser)
data = data.select( \
  "CAST(sepal_length AS DOUBLE) AS sepal_length, "
  + "CAST(sepal_width AS DOUBLE) AS sepal_width, "
  + "CAST(petal_length AS DOUBLE) AS petal_length, "
  + "CAST(petal_width AS DOUBLE) AS petal_width, category”)
```

加载逻辑回归模型，对流数据进行预测

```
model = CsvSourceBatchOp().setFilePath("/path/to/model.csv") \
  .setSchemaStr("model_id bigint, model_info string, label_type
string")

lr_predictor = LogisticRegressionPredictStreamOp(model) \
  .setPredictionCol("pred").setPredictionDetailCol("pred_detail")

result = data.link(lr_predictor)
```

将预测结果写出到kafka

```
sink = KafkaSinkStreamOp() \
  .setBootstrapServers("localhost:9092") \
  .setDataFormat("json").setTopic("lr_pred")
data.link(sink)
StreamOperator.execute()
```

Alink version 1.1.1

- 数据列参数、枚举类型参数的检验和提示
- 优化Alink批式组件与Python Dataframe之间数据转换的速度
- 当useRemoveEnv时自动检测localIp
- 新增组件，将CSV、JSON和KV格式的字符串解析为多列
- 新增组件WindowGroupByStreamOp，简化流式数据的窗口分组操作

优化枚举类型参数

SelectorType可以填写NumTopFeatures, Percentil, FPR等，是枚举类型变量

```
selector = ChiSqSelectorBatchOp()\
            .setSelectorType("aaa")\
            .setSelectedCols(["f_string", "f_long", "f_int", "f_double"])\
            .setLabelCol("f_boolean")\
            .setNumTopFeatures(2)
```

SelectorType输出错误的值AAA，异常信息不明显，没有指出是哪个参数写错了

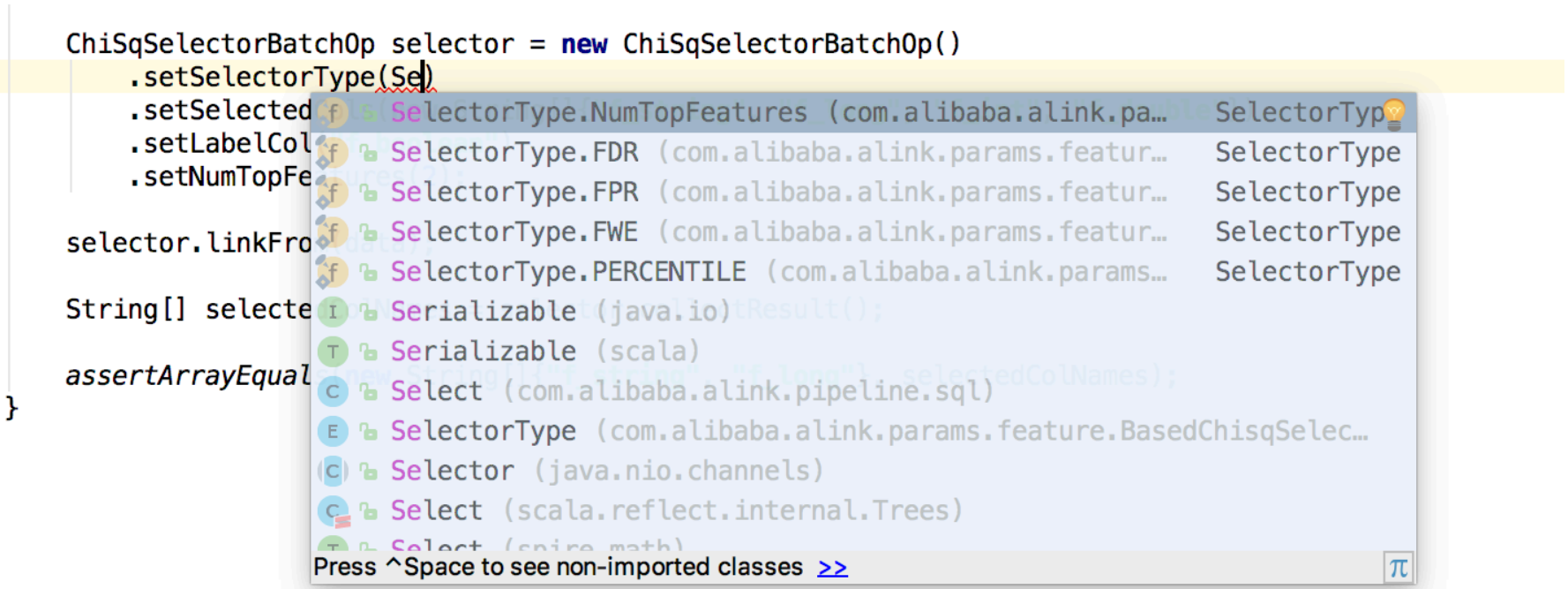
```
Caused by: java.lang.IllegalArgumentException: No enum constant com.alibaba.alink.operator.common.statistics.ChiSquar
eTest$ChiSqSelectorType.AAA
        at java.lang.Enum.valueOf(Enum.java:238)
        at com.alibaba.alink.operator.common.statistics.ChiSquareTest$ChiSqSelectorType.valueOf(Unknown Source)
```

1.1.1版本优化之后, 异常信息中会有哪个参数填写错误，和值可能是什么。

```
py4j.protocol.Py4JJavaError: An error occurred while calling o26.setSelectorType.
: java.lang.RuntimeException: aaa is not member of selectorType.It maybe NumTopFeatures,PERCENTILE,FPR,FDR,FWE.
        at com.alibaba.alink.params.ParamUtil.searchEnum(ParamUtil.java:68)
        at com.alibaba.alink.params.ParamUtil.searchEnum(ParamUtil.java:18)
        at com.alibaba.alink.params.ParamUtil.searchEnum(ParamUtil.java:29)
        at com.alibaba.alink.params.feature.BasedChisqSelectorParams.setSelectorType(BasedChisqSelectorParams.java:73)
```

优化枚举类型参数

如果使用JAVA，还会有输入的提示



优化列名参数

算法中会有很多列名参数，列名不存在情况也很常见，

```
# -*- coding=UTF-8 -*-
data = np.array([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969']
])

df = pd.DataFrame({"id": data[:, 0], "text": data[:, 1]})
inOp = dataframeToOperator(df, schemaStr='id long, text string', op_type='batch')

segment = SegmentBatchOp()\
            .setSelectedCol('text1')\
            .setOutputCol("segment")

BatchOperator.collectToDataframe(segment.linkFrom(inOp))
```

在1.1.1版本里，不仅抛出哪列不存在，也会提示最可能的列名，帮助用户做判断。

```
Caused by: java.lang.IllegalArgumentException: Can not find column: text1, do you mean: text ?
        at com.alibaba.alink.common.utils.TableUtil.findColIndexWithAssertAndHint(TableUtil.java:98)
        at com.alibaba.alink.common.mapper.SISOColsHelper.<init>(SISOColsHelper.java:47)
        at com.alibaba.alink.common.mapper.SISOMapper.<init>(SISOMapper.java:26)
        at com.alibaba.alink.operator.common.nlp.SegmentMapper.<init>(SegmentMapper.java:22)
        at com.alibaba.alink.operator.batch.utils.MapBatchOp.linkFrom(MapBatchOp.java:34)
        ... 11 more
```

优化列名参数

Java的行为相同

```
Row[] rows = new Row[] {
    Row.of(1, "别人复习是查漏补缺")
};

MemSourceBatchOp data = new MemSourceBatchOp(rows, new String[] {"id", "text"});

Segment op = new Segment()
    .setSelectedCol("text1")
    .setOutputCol("output");

BatchOperator res = op.transform(data);
```

在1.1.1版本里，不仅抛出哪列不存在，也会提示最可能的列名，帮助用户做判断。

```
Caused by: java.lang.IllegalArgumentException: Can not find column: text1, do you mean: text ?
    at com.alibaba.alink.common.utils.TableUtil.findColIndexWithAssertAndHint(TableUtil.java:95)
    at com.alibaba.alink.common.mapper.SISOColsHelper.<init>(SISOColsHelper.java:47)
    at com.alibaba.alink.common.mapper.SISOMapper.<init>(SISOMapper.java:26)
    at com.alibaba.alink.operator.common.nlp.SegmentMapper.<init>(SegmentMapper.java:22)
    at com.alibaba.alink.operator.batch.utils.MapBatchOp.linkFrom(MapBatchOp.java:34)
    ... 27 more
```

PyAlink 1.1.1 的改进

- 优化了 DataFrame 和 BatchOperator 互转的性能
  - 右侧代码是将一个行数较多的 DataFrame 转为 BatchOperator 的示例代码。之前 50000 行数据需要约 55s，现在只需要 5s；100w 行数据约 20s。
  - collectToDataframe 同样进行了优化。

```
n = 50000
users = []
for col in range(n):
    users.append([col] * 2)
df = pd.DataFrame(users)
source = BatchOperator\
    .fromDataframe(df, schemaStr='id int, label int')
source.link(
    CsvSinkBatchOp()
        .setOverwriteSink(True)
        .setFilePath('temp.csv')
)
BatchOperator.execute()
```

- 使用 流式组件的 print 功能时，将不会因为数据中有 NaN 导致作业失败。

PyAlink 1.1.1 的改进

- Python UDF 运行中将自动检测 python3 命令
  - 之前版本中，如果环境中同时有 Python 2 和 3，将可能因为 python 命令指向 Python 2 而导致运行不成功。
  - 现在将优先 python3 来执行 Python UDF。

- useRemoteEnv 将自动检测本机外网 IP
  - 在一般网络配置下，使用 StreamOperator 组件的 print 功能无需手动设置 localIp 了。

useRemoteEnv(host, port, parallelism, flinkHome=None, localIp=None, shipAlinkAlgoJar=True, config=None)

新增组件，将CSV、JSON和KV格式的字符串解析为多列

{"sepal_width":3.4,"petal_width":0.2,"sepal_length":4.8,"category":"Iris-setosa","petal_length":1.6}
{"sepal_width":4.1,"petal_width":0.1,"sepal_length":5.2,"category":"Iris-setosa","petal_length":1.5}
{"sepal_width":2.8,"petal_width":1.5,"sepal_length":6.5,"category":"Iris-versicolor","petal_length":4.6}
{"sepal_width":3.0,"petal_width":1.8,"sepal_length":6.1,"category":"Iris-virginica","petal_length":4.9}
{"sepal_width":2.9,"petal_width":1.8,"sepal_length":7.3,"category":"Iris-virginica","petal_length":6.3}

sepal_length	sepal_width	petal_length	petal_width	category
4.8000	3.4000	1.6000	0.2000	Iris-setosa
5.2000	4.1000	1.5000	0.1000	Iris-setosa
6.5000	2.8000	4.6000	1.5000	Iris-versicolor
6.1000	3.0000	4.9000	1.8000	Iris-virginica
7.3000	2.9000	6.3000	1.8000	Iris-virginica

```
json_parser = JsonValueStreamOp()
        .setSelectedCol("message")
        .setOutputCols(["sepal_length", "sepal_width", "petal_length",
"petal_width", "category"])
        .setJsonPath(["$.sepal_length", "$.sepal_width", "$.petal_length",
"$.petal_width", "$.category"])

data = data.link(json_parser)
data = data.select( \
  "CAST(sepal_length AS DOUBLE) AS sepal_length, "
  + "CAST(sepal_width AS DOUBLE) AS sepal_width, "
  + "CAST(petal_length AS DOUBLE) AS petal_length, "
  + "CAST(petal_width AS DOUBLE) AS petal_width, category")
```

```
data = data.link(\
JsonToColumnsStreamOp()
.setSelectedCol("message")
.setSchemaStr("sepal_length double, sepal_width double, petal_length double, "
        + "petal_width double, category string")
.setReservedCols( [] )
)
```

新增组件，将CSV、JSON和KV格式的字符串解析为多列

```
66.249.79.35 - - [14/Jun/2018:06:45:24 +0000] "GET /img/20180504/702434-20180302101540805-554506523.jpg HTTP/1.1" 200 10013 "-" "Googlebot-Image/1.0"
66.249.79.35 - - [14/Jun/2018:06:45:25 +0000] "GET /img/20180504/702434-20180302161346635-1714710787.jpg HTTP/1.1" 200 45157 "-" "Googlebot-Image/1.0"
66.249.79.35 - - [14/Jun/2018:06:45:56 +0000] "GET /img/2018/05/21/60662344.jpg HTTP/1.1" 200 14133 "-" "Googlebot-Image/1.0"
54.36.148.129 - - [14/Jun/2018:06:46:01 +0000] "GET /archives/91007 HTTP/1.1" 200 8332 "-" "Mozilla/5.0 (compatible; AhrefsBot/5.2; +http://ahrefs.com/robot/)"
54.36.148.201 - - [14/Jun/2018:06:46:03 +0000] "GET /archives/88741/feed HTTP/1.1" 200 983 "-" "Mozilla/5.0 (compatible; AhrefsBot/5.2; +http://ahrefs.com/robot/)"
5.255.250.200 - - [14/Jun/2018:06:46:03 +0000] "GET /archives/87084 HTTP/1.1" 200 9951 "-" "Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots)"
```

```
source
    .select(
        "SUBSTRING( text FROM 1 For POSITION('[' IN text)-2 ) AS part1, "
            + "REGEXP_EXTRACT( text, '(\\[)(.*?)(\\])', 2 ) AS log_time,"
            + "SUBSTRING(text FROM 2+POSITION(']' IN text)) AS part2"
    )
    .lazyPrint( n: 5,  title: "\n### first log split ###")
    .link(
        new CsvToColumnsBatchOp()
            .setSelectedCol("part1")
            .setFieldDelimiter(" ")
            .setSchemaStr("ip string, col1 string, col2 string")
    )
    .link(
        new CsvToColumnsBatchOp()
            .setSelectedCol("part2")
            .setFieldDelimiter(" ")
            .setSchemaStr("cmd string, response int, bytesize int, col3 string, col4 string")
    )
    .link(
        new CsvToColumnsBatchOp()
            .setSelectedCol("cmd")
            .setFieldDelimiter(" ")
            .setSchemaStr("req_method string, url string, protocol string")
    )
    .select("ip, col1, col2, log_time, req_method, url, protocol, response, bytesize, col3, col4")
```

Alink相关材料

https://github.com/alibaba/Alink

批式CSV数据读取【Alink使用技巧】
Alink如何读写Libsvm格式数据【Alink使用技巧】
Alink如何读写文本数据【Alink使用技巧】
Alink连接Kafka数据源（Java版本）
Alink连接Kafka数据源（Python版本）
Alink批式数据转化为Python的DataFrame【Alink使用技巧】
Python数组如何转化为批式数据源？【Alink使用技巧】

Alink LocalPredictor简介

Alink中文情感分析示例（Java版本）
Alink中文情感分析示例（Python版本）

Alink在线学习(Online Learning)示例【一】
Alink在线学习(Online Learning)之Java示例【一】

Apache Flink

THANKS

阿里计算平台Alink开源用户群

点击提交申请后自动加入该群