特征预处理

## 算子简介

特征预处理算子的目的是为spark的数据框提供一个特征提取、特征转换、特征选择等（以下统一称特征预处理）的接口，方便后面spark.ml或spark.mllib调用。

该算子主要封装spark.ml.feature中的特征预处理的API（目前截止到spark 2.3）为一个算子，并添加一些字段属性判断、缺失值过滤等功能。

## 特征提取

由于本算子集成了较多的算法，每种算法要求的输入数据格式、参数和输出格式都不同，本节对算子中的各个模块进行简介。

1. 属性类特征提取
2. 正则表达式分词

实现的是ml.feature中的

预处理列：此时需要类型为String类型

输出列：输出一个ArrayType(StringType,true)类型的列

匹配模式：请输出合适的正则表达式模式。如果正则表达式不正确，限于spark的lazy模式，目前只能捕捉非执行过程的异常。

是以此为分隔还是以此为匹配类型：如果选择以此为分隔，则算子会根据正则表达式找到对应的字符串并以此为分隔符对整个字符串进行分割，否则则会遍历整个字符串找到对应的子串。

是否转为小写：会将预处理列中的字符串中的英文转为小写

1. 词频统计

根据运行的逻辑可以将算法拆分为两个阶段：训练和预测，训练是指训练一个分词模型（主要是根据设定参数训练一个带顺序的词汇表），预测是根据分词模型将输入数据进行频次统计后转为Vector类型。

**参数说明：**

预处理列：ArrayType(StringType,true)类型的列

获取训练模型：分为从持久化引擎中获取词频模型和训练新模型两种

1. 从持久化引擎中获取模型：

输入模型路径

输入预测表输入列名

1. 训练新模型

训练表名：训练表的名称

训练列名：ArrayType(StringType,true)类型的列

最大词汇数，系统会根据所有词汇的词频取取最常出现的n个，这里n就是最大词汇数，默认值是2^18。可以根据数据设置合适的大小，建议不要太大，该算法中需要将所有的词汇表collect到driver端，过大的词汇数会导致总词汇的内存超过akka.frameSize的上界。

最小文档频率，当某词汇出现在文档中的频率低于该频率时会被从词汇表中清除掉。

是否持久化保存新的训练模型：是/否，如果选择是则需要输入保存路径。

最小词频：某条记录的词汇中低于此词频数的词汇会被过滤掉。

预测列名：输出mllib.linalg.Vector类型的列，是由数值类型组成的向量

1. Hash词频统计

该算法对每个词汇采用，hash并将hash值通过取模的方式映射到指定索引（从0到（长度-1））。不同于词频统计，此处没有将每个词汇collect获得词汇表再统计词频，而是建立一种由词汇到索引映射关系再获得词频向量，因而避免了获取词汇表时的结点通信。缺点是，不同词汇的hash值可能相等，取模时不同的hash可能相等，庆幸的是当我们将参数调整到合理的值时，这种概率是很小的。

**参数说明：**

输入列名：ArrayType(StringType,true)类型的列

词汇数：默认值是2^18，需要小于2^31

输出列名：Vector

1. Word2Vec

Word2vec是现在自然语言处理中常用语义分析的一种技术，它采用一系列代表文档的词语来训练word2vecmodel。该模型将每个词语映射到一个固定大小的向量。word2vecmodel使用文档中每个词语的平均数来将文档转换为向量，然后这个向量可以作为预测的特征，来计算文档相似度计算等等。常见的Word2Vec模型包括：Skip-gram、CBOW等，目前spark API实现的是Skip-gram。

Skip-gram本质上是一个自学习的神经网络（自编码器），与一般神经网络不同的是这里的隐含层没有激活函数，而只是在输出层使用了激活函数。因此在输出层之前的隐含层实现的就是一个由（词汇数\*1）到（隐藏层\*1）的一个线性映射，这里隐藏层等于向量长度，也就是乘以一个维度为（词汇数\*向量长度）的矩阵。

**参数说明：**

预处理列：ArrayType(StringType,true)类型的列

获取训练模型：分为从持久化引擎中获取词频模型和训练新模型两种

1. 从持久化引擎中获取模型：

输入模型路径

输入预测表输入列名：ArrayType(StringType,true)类型的列

1. 训练新模型

训练表名：训练表的名称

训练列名：ArrayType(StringType,true)类型的列

最大词汇数：系统会根据所有词汇的词频取取最常出现的n个，这里n就是最大词汇数，默认值是2^18。可以根据数据设置合适的大小，建议不要太大，该算法中需要将所有的词汇表collect到driver端，过大的词汇数会导致总词汇的内存超过akka.frameSize的上界。

最小文档频率，当某词汇出现在文档中的频率低于该频率时会被从词汇表中清除掉。

是否持久化保存新的训练模型：是/否，如果选择是则需要输入保存路径。

最小词频：某条记录的词汇中低于此词频数的词汇会被过滤掉，这里说的是所有记录中的词频，通过最小词频可以去掉一些频数比较小的词频，降噪，同时还可以降低词汇表的数目，减少集群间数据传输的压力（可参见本文算法部分“参数使用注意事项3”）。

向量长度 默认值为100

学习率 默认值为0.025

最低词频 默认值为5

NumPartition 并行度 默认值为1，即不并行。

这里并行度等于设定的分区数，对每个分区的词汇分别进行训练得到对应系数最后在进行整合。因此在集群还有空闲运算资源的情况下，并行度越大运算越快，但模型越不精确。

在集群能够运行的情况下可以设小一些以求得到最大精度。

循环次数：将模型训练多少次，每次训练都会使用上次迭代结果作为初值。默认值为1，即循环一次即终止。

预测列名：输出mllib.linalg.Vector类型的列，是由数值类型组成的向量

**参数使用注意事项：**

基于以上原理我们在模型是需要注意：

1. 训练数据的数目需要大于模型数——词汇数目\*向量长度
2. 词汇数目\*向量长度最大不能超过2^31 / 8 = 2^28
3. 为了akka.frameSize超界（spark的API会将所有词汇的频次统计collect到driver端）等问题同时也为了增加模型的自由度，最好确保总词汇数目不要过大。可以先将词汇进行标准化，比如现将词汇去掉停用词、词汇去掉数字等，以降低词汇数目，您还可以通过调大最低词频来降低总的词汇数目。
4. 停用词移除

在自然语言中，很多语句中的词汇在进行训练时对我们用处不大，比如在进行语义分析时一些停用词如“具体来说，以下分为三个方面：1）时间发生了变化；2）地方发生了变化”。在一定假设条件下 “具体来说”可以作为停用词，因为我们可以认为它只是语气意义上的延续。

**参数说明：**

输入数据：要处理的表名。

输入预处理列名：ArrayType(StringType,true)类型的列

请选择停用词模式：可以选择“汉语”、“英语”、“手动输入”，“从hdfs集群中读取”四种模式。其中“汉语”停用词收录了互联网上整理的哈工大停用词词库1872个停用词，“英语”停用词收录了“Glasgow Information Retrieval Group”贡献的318个停用词。“手动输入”是以动态列表的形式输入，从“hdfs集群中读取”，需要提供路径和分隔符。

是否忽略大小写：如果选择是，会不计大小写过滤掉停用词。

输入输出列名：输出一个ArrayType(StringType,true)类型的列

1. n-gram

即自然语言处理中常用的ngram方法，本质就是通过一个定长的平滑窗口，对一个记录依次进行取值，从而获得一个考虑到前后文的词语特征。例如一个[“我”，“听说”，“小明”，“中了”，“彩票”]，如果采用窗口为2的n-gram方法进行特征转换，会得到[“我 听说”，“听说 小明”，“小明 中了”，“中了 彩票”]。窗口数越大获得的上下文信息越多，不过词汇数也因此增加，需要结合后面用到的机器学习算法设定恰当的gram窗口长度。

**参数说明：**

输入数据：要处理的表名。

输入预处理列名：ArrayType(StringType,true)类型的列

请输入gram窗宽：输入一个大于1的整数。

输入输出列名：输出一个ArrayType(StringType,true)类型的列

1. 数值类型特征提取
2. 数值数据离散化

将数值类型的数据通过分箱形式离散化，返回所在箱子的索引。

输入数据：要处理的表名。

输入预处理列名：ArrayType(StringType,true)类型的列

请选择分箱模式：等宽分箱、等深分箱、自定义边界分箱

1. 等宽分箱

周期初始相位——某个箱子的起始数值：输入合理的初始值。可以是任意箱子的起始值。

箱子宽度：输入正数。

1. 等深分箱

以深度方式设定分箱、以箱子数方式设定分箱

以深度方式设定分箱：

箱子深度：大于0的整数，同时

以箱子数设定分箱

箱子数：

1. 自定义边界分箱

是否忽略大小写：如果选择是，会不计大小写过滤掉停用词。

输入输出列名：输出一个ArrayType(StringType,true)类型的列

1. 正则分词器

表达式或者分隔符

是否以此作为分隔符。

1. 说明

特征提取和特征转换只是基于应用产生的概念，因此单纯就方法来说可能有些方法既可以作为特征提取也可以作为特征转换（比如分析器）。该算子的划分规则为：如果将某个混合在一起无法直接使用的信息提取成向量或其他可用形式我们认为是特征提取（比如分词），单纯的进行一些数值上或者字符上的变换我们认为是特征转换（比如词频统计、hash，tf-idf转换等等）。

1. 特征转换
2. tf-idf

词频－逆向文件频率（TF-IDF）是一种在文本挖掘中广泛使用的特征向量化方法，它采用了频率乘以特异性的模式，可以兼顾一个文档中词语在语料库中的稀有程度和词频两方面的特征。

特异性主要用逆文档频率来衡量，如果以D代表整个语料库，以t代表词，以d代表文档，则tf-idf变换如下：



总的tf-idf公式就是：



注意：

1. 该算子认为一个DataFrame是一个整个语料库
2. 由于最后是以稀疏向量的形式输出的，该算子需要输入一个提取的特征数，以防止栈溢出（由于需要将整个特征词放到driver端），该数目因集群而异。另外结合
3. 此处采用hashingTF，此处没有对每个word建立index->word的映射，而是采用“hash+模”的方式建立了映射，此处需要设定一个参数——词汇数，将每个词映射到0到词汇数中，默认词汇数为2^20，开发者建议在2^18到2^20之间。可见：两个不同的词可能会被统计到一个词中；上述时间默认的特征数越大发生的概率越低，但由于hash函数的特性始终不能避免。
4. 多项式展开

常用语图像等信号的识别时的特征转换，有助于识别具备非线性特征的数据。

1. 离散余弦变换

是将时域的N维实数序列转换成频域的N维实数序列的过程（有点类似离散傅里叶变换）。

离散余弦变换是与傅里叶变换相关的一种变换，它类似于离散傅立叶变换但是只使用实数。离散余弦变换相当于一个长度大概是它两倍的离散傅里叶变换，这个离散傅里叶变换是对一个实偶函数进行的（因为一个实偶函数的傅里叶变换仍然是一个实偶函数）。离散余弦变换，经常被信号处理和图像处理使用，用于对信号和图像（包括静止图像和运动图像）进行有损数据压缩。

1. 特征选择

具体的应用API如下表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 特征提取 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 算子简介

## 算子简介

## 算子简介

## 算子简介