商品关键词的词向量计算(word2vec)

word2vec算法可以计算出每个词语的一个词向量，我们可以用它来表示该词的语义层面的含义

In [1]:

```
import os
# 配置pyspark和spark driver运行时 使用的python解释器
JAVA_HOME = '/root/bigdata/jdk'
PYSPARK_PYTHON = '/miniconda2/envs/py365/bin/python'
# 当存在多个版本时，不指定很可能会导致出错
os.environ['PYSPARK_PYTHON'] = PYSPARK_PYTHON
os.environ['PYSPARK_DRIVER_PYTHON'] = PYSPARK_PYTHON
os.environ['JAVA_HOME'] = JAVA_HOME
# 配置spark信息
from pyspark import SparkConf
from pyspark.sql import SparkSession

SPARK_APP_NAME = "SKUSimilarity"
SPARK_URL = "spark://192.168.58.100:7077"

conf = SparkConf()    # 创建spark config对象
config = (
    ("spark.app.name", SPARK_APP_NAME),    # 设置启动的spark的app名称，没有提供，将随
    ("spark.executor.memory", "2g"),    # 设置该app启动时占用的内存用量，默认1g，指一
    ("spark.master", SPARK_URL),    # spark master的地址
    ("spark.executor.cores", "2"),    # 设置spark executor使用的CPU核心数，指一台虚拟
    ("hive.metastore.uris", "thrift://localhost:9083"),    # 配置hive元数据的访问，否
)
# 查看更详细配置及说明：https://spark.apache.org/docs/latest/configuration.html

conf.setAll(config)

# 利用config对象，创建spark session
spark = SparkSession.builder.config(conf=conf).enableHiveSupport().getOrCreate()
```

In [4]:
```
sku_detail = spark.sql('select * from sku_detail')
electronic_product = sku_detail.where('category1_id<6 and category1_id>0')
from pyspark.sql.functions import concat_ws
sentence_df = electronic_product.select('sku_id','category1_id',\
            concat_ws(',',\
                      electronic_product.category1,\
                      electronic_product.category2,\
                      electronic_product.category3,\
                       electronic_product.name,\
                       electronic_product.caption,\
                       electronic_product.price,\
                       electronic_product.specification
                      ).alias('summary')
)
```

In [5]:
```
sentence_df.show()
```

```
+------+-----------+--------------------+
|sku_id|category1_id|             summary|
+------+-----------+--------------------+
|   148|          3|数码,数码配件,读卡器,随身厅 W...|
|   463|          3|数码,数码配件,读卡器,飞花令 安...|
|   471|          3|数码,数码配件,读卡器,【包邮】飞...|
|   496|          3|数码,数码配件,读卡器,品胜（PI...|
|   833|          3|数码,数码配件,读卡器,LEXAR...|
|  1088|          2|相机,摄影摄像,数码相框,青美 壁...|
|  1238|          3|数码,数码配件,读卡器,dypla...|
|  1342|          3|数码,数码配件,读卡器,绿联（UG...|
|  1580|          2|相机,摄影摄像,数码相框,HNM ...|
|  1591|          3|数码,数码配件,读卡器,kisdi...|
|  1645|          2|相机,摄影摄像,数码相框,爱国者（...|
|  1829|          3|数码,数码配件,读卡器,金士顿（K...|
|  1959|          2|相机,摄影摄像,数码相机,理光（R...|
|  2122|          1|手机,手机配件,移动电源,贝视特苹...|
|  2142|          1|手机,手机配件,移动电源,戈派 无...|
|  2366|          1|手机,手机配件,移动电源,赋电 充...|
|  2659|          1|手机,手机配件,移动电源,OISL...|
|  2866|          1|手机,手机通讯,对讲机,宝锋（BA...|
|  3175|          1|手机,手机通讯,对讲机,Motor...|
|  3749|          1|手机,手机通讯,对讲机,ZASTO...|
+------+-----------+--------------------+
only showing top 20 rows
```

In [38]:
```
sentence_df.count()
```

Out[38]: 66651

分词

首先处理电子产品

In [10]:

```
def words(partitions):

    import os

    import jieba
    import jieba.analyse
    import jieba.posseg as pseg
    import codecs

    abspath = "/root/workspace/3.rs_project/project2/notebook"

    stopwords_path = os.path.join(abspath, 'keywordExtract/extract/baidu_stopwords.tx

    # 结巴加载用户词典
    userDict_path = os.path.join(abspath, "keywordExtract/extract/词典/all.txt")
    jieba.load_userdict(userDict_path)

    # 停用词文本
    stopwords_path = os.path.join(abspath, "keywordExtract/extract/baidu_stopwords.tx


    def get_stopwords_list():
        """返回stopwords列表"""
        stopwords_list = [i.strip()
                          for i in codecs.open(stopwords_path).readlines()]
        return stopwords_list

    # 所有的停用词列表
    stopwords_list = get_stopwords_list()

    def cut_sentence(sentence):
        """对切割之后的词语进行过滤，去除停用词，保留名词，英文和自定义词库中的词，长
        # print(sentence,"*"*100)
        # eg:[pair('今天', 't'), pair('有', 'd'), pair('雾', 'n'), pair('霾', 'g')]
        seg_list = pseg.lcut(sentence)
        seg_list = [i for i in seg_list if i.flag not in stopwords_list]
        filtered_words_list = []
        for seg in seg_list:
            # print(seg)
            if len(seg.word) <= 1:
                continue
            elif seg.flag == "eng":
                if len(seg.word) <= 2:
                    continue
                else:
                    filtered_words_list.append(seg.word)
            elif seg.flag.startswith("n"):
                filtered_words_list.append(seg.word)
            elif seg.flag in ["x", "eng"]:  # 是自定一个词语或者是英文单词
                filtered_words_list.append(seg.word)
        return filtered_words_list

    for row in partitions:
        yield (cut_sentence(row.summary),)
doc = sentence_df.rdd.mapPartitions(words)
doc = doc.toDF(['words'])
```

```
57 doc
```

Out[10]: DataFrame[words: array<string>]

In [13]:
```
1 doc.show(5,truncate=False)
```

```
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------+
|words
|
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------+
|[数码, 数码配件, 读卡器, WPOS, 高度, 业务, 智能, 终端, 森锐, 触摸屏, 收银机, 身份, 包
邮, 正品, 购物]
|
|[数码, 数码配件, 读卡器, 飞花, 安卓, 手机, 读卡器, Type, USB, OTG, 车载, 读卡器, 转接
器, 内存卡, 读卡器, 黑色, 私人, 联系, 客服, 型号, 效果图, 型号, 颜色, Type, 读卡器, 金
色, 颜色, 安卓, 手机, 电脑, 读卡器, 蓝色, 颜色, 安卓, 电脑, 黑色, 颜色, 安卓, 电脑, typ
e, 颜色, 安卓, 电脑, type, 颜色, 安卓, 手机, 读卡器, Mirco, 金色, 颜色, 手机, 读卡器,
金色, 颜色, 电脑, USB3, Type, 黑色, 颜色, 内存卡, 读卡器, 黑色, 颜色, 闪迪, 卡套]
|
|[数码, 数码配件, 读卡器, 包邮, 飞花, 安卓, 外置, 手机, 读卡器, 手机, 内存卡, MicroSD,
读卡器, 于安卓, 手机, 安卓, 手机, 读卡器, 金色, vivoY66, Y67, Y51A, Y31, micro, usb, 读
卡器, 金属外壳, 材质, 挂链, 版本, N4S, N4A, f4s, 版本, vizza, n6pro, N5S, 版本, OPPO, A
79, A77, A73, 版本, OPPOR11, R11Plus, r11t, a77, 版本, OPPOR9s, R9sPlus, R9Plus, 版本,
VIVO, y81s, y97, 版本, oppo, r15, a7X, R15X, 版本, vivo, X9Plus, X7Plus, 版本, vivo, Xp
lay6, Xplay5A, 版本, vivo, x23, z1i, Z3i, 版本, vivo, y69, y71, y75, y79, y85, 版本, vi
voY66, Y67, Y51A, Y31, 版本, 华为, Mate8, 华为, Mate7, 版本, 华为, 荣耀, max, 华为, 版
本, 华为, plus, 版本, 华为, plus, max, 版本, 华为, plus, 版本, 华为, 荣耀, 青春, 版本,
华为, 荣耀, 版本, 安卓, 手机, 手机, OTG, 功能, 版本, 小米, 红米, 红米, 红米, 版本, 红
米, note2, 版本, 红米, pro, note5, 版本, 红米, note4X, 红米, note3, 版本, 红米, note5a,
红米, Plus, 版本, 荣耀, 版本, 金立, S10, S10C, S10B, 版本, 魅族, 魅族, note8, 版本, 魅
族, 魅蓝, Note5, 魅蓝, 颜色, 安卓, 手机, 读卡器, 金色]|
|[数码, 数码配件, 读卡器, 品胜, PISEN, 全能, 读卡器, usb, 手机, 带线, 全能王, 台式机,
京东, 理由, 退换货, 品牌, 信赖, 颜色, 读卡器, 颜色, 读卡器, 颜色, 全能王, 笔记本, 颜色,
读卡器, 颜色, 全能王, 台式机]
|
|[数码, 数码配件, 读卡器, LEXAR, 雷克沙, Lexar, USB, 读卡器, 读卡器, 读卡器, 版本, 读卡
器]
|
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

```
+----------------------------------+----------------------------------+----------------------------------+----------------------------------+----------------------------------+
+----------------------------------+----------------------------------+----------------------------------+----------------------------------+----------------------------------+
+----------------------------------+----------------------------------+----------------------------------+----------------------------------+----------------------------------+
+----------------------------------+----------------------------------+----------------------------------+----------------------------------+----------------------------------+
----------------------------------+
only showing top 5 rows
```

In [36]:
```
1 doc.count()
```

Out[36]: 66651

word2vec模型训练

In [15]:
```
1 from pyspark.ml.feature import Word2Vec
2 # vectorSize： Word2Vec训练得到的向量 维度是100
3 word2Vec = Word2Vec(vectorSize=100,inputCol='words',outputCol='model')
4 model = word2Vec.fit(doc)
```

In [21]:
```
1 from pyspark.sql.functions import format_number as fmt
2 # findSynonyms("笔记本", 20):Find "20" number of words closest in similarity to "笔记
3 # 四舍六入保持小数点后5位
4 model.findSynonyms("笔记本", 20).select("word", fmt("similarity", 5).alias("similarity
5 model.findSynonyms("荣耀", 20).show()
```

...

In [22]:
```
1 # model.save('/meiduo_mall/models/电子产品.word2vec_model')
```

In [23]:
```
1 from pyspark.ml.feature import Word2VecModel
2 model=Word2VecModel.load('/meiduo_mall/models/电子产品.word2vec_model')
```

In [29]:
```
vectors = model.getVectors()
# head对于dataframe 类似  take对于rdd
vectors.head(100)
```

Out[29]:
```
[Row(word='钟爱', vector=DenseVector([0.0273, 0.0457, -0.055, -0.0155, -0.0001, 0.0014, 0.0935, -0.0592, 0.0865, -0.0916, -0.0288, 0.1004, -0.0364, 0.0164, 0.0715, 0.0035, -0.0009, -0.0474, 0.0531, -0.077, -0.0859, -0.0244, 0.103, -0.0842, -0.032, 0.0565, 0.0002, 0.0855, 0.0344, -0.0066, -0.0757, 0.0414, 0.0119, 0.0671, 0.0794, 0.0482, -0.0299, -0.0478, -0.1022, 0.0813, -0.1129, 0.0368, -0.0284, -0.0803, -0.0222, 0.0714, -0.0212, 0.0656, 0.0207, -0.1059, -0.0181, -0.1638, -0.039, 0.0062, -0.0052, -0.0536, -0.063, -0.0101, 0.0072, -0.0572, 0.0102, -0.0392, 0.0023, 0.0344, -0.0152, 0.0213, -0.0483, 0.1004, -0.0395, 0.0414, -0.0138, -0.0225, 0.03, -0.0638, -0.0778, -0.0217, -0.1213, -0.012, -0.0017, 0.0308, 0.0865, -0.0251, -0.0385, 0.0312, -0.0577, 0.0681, -0.0561, -0.1156, 0.0054, -0.0154, -0.0829, -0.1159, 0.0046, 0.0634, 0.0079, -0.0352, -0.0339, -0.1092, 0.0396, 0.0603])),
 Row(word='伙伴', vector=DenseVector([0.0182, 0.1059, -0.0912, -0.3366, -0.0724, -0.0526, 0.1929, -0.0478, 0.1557, 0.0432, 0.2573, 0.1979, -0.0803, 0.0441, 0.1312, -0.086, -0.0814, -0.1044, -0.1085, 0.0818, -0.0301, 0.0156, -0.0508, -0.1719, -0.2242, 0.19, 0.0292, 0.1213, 0.0973, -0.0332, 0.0316, -0.1109, 0.0583, -0.0871, 0.1265, 0.1263, 0.1059, 0.1125, -0.0749, 0.0025, -0.2219, -0.0244, -0.175, -0.0109, -0.0175, 0.0993, 0.1881, 0.2325, -0.1345, -0.0046, 0.0639, -0.0861, 0.0167, 0.1498, 0.0121, -0.1282, -0.113, 0.0565, 0.0039, 0.0354, 0.0979, -0.2051, 0.2172, 0.091, -0.0627, -0.0183, 0.0329, 0.102, -0.0223, 0.2082, -0.1045, 0.0277, 0.0951, 0.0557, -0.1943, 0.0206, 0.082,
```

In [35]:
```
# 证明 训练模型使用的Word2Vec(vectorSize=100,inputCol='words',outputCol='model')中的ve
len(vectors.head(10)[1].vector.tolist())
```

Out[35]: 100

In [37]:
```
# 电子产品中 所有的sku的 使用embedding向量表示的关键词 总数(w2v训练会自动去掉很多关键词
vectors.count()
```

Out[37]: 18121