Examen 2

175904 - Jorge III Altamirano Astorga

Carga de los Datos

In [1]:
```
import pyspark
from pyspark import SparkContext, SparkConf, SQLContext
from pyspark.sql.functions import *
from pyspark.sql import *
from pyspark.sql.types import *
import time, os, re, glob, sys
# https://spark.apache.org/docs/latest/configuration.html
conf = SparkConf()
conf.set("spark.worker.cleanup.appDataTtl", 24*60*60)
conf.set("spark.worker.cleanup.enabled", True)
conf.set("spark.driver.memory", "60g")
conf.set("spark.driver.cores", 5)
conf.set("spark.driver.memoryOverhead", 0.9)
conf.set("spark.executor.memory", "60g")
conf.set("spark.executor.cores", 5)
conf.set("spark.jars", "file:/usr/local/spark-2.3.0-bin-hadoop2.7/jars/spark-nlp
_2.11-1.5.3.jar," +
         "file:/usr/local/spark-2.3.0-bin-hadoop2.7/jars/config-1.3.0.jar," + #n
eeded nlp
         "local:/usr/local/spark-2.3.0-bin-hadoop2.7/jars/hadoop-common-2.7.3.ja
r," + #needed by aws
         "local:/usr/local/spark-2.3.0-bin-hadoop2.7/jars/commons-cli-1.2.jar,"
+ #needed by aws
         "file:/usr/local/spark-2.3.0-bin-hadoop2.7/jars/hadoop-aws-2.7.3.jar,"
+ #needed by aws
         "file:/usr/local/spark-2.3.0-bin-hadoop2.7/jars/aws-java-sdk-1.7.4.jar"
) #needed by aws
conf.set("spark.jars.packages", "JohnSnowLabs:spark-nlp:1.5.3")
conf.set("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem")
### get they creds to login to AWS :-)
HOME = os.environ["HOME"]
aws_id, aws_key = (None, None)
with open(HOME+"/.aws/credentials", "r") as f:
    for line in f:
        line = line.strip()
        if "aws_access_key_id" in line:
            aws_id = re.sub("^.*aws_access_key_id\s*=\s*", "", line)
        elif "aws_secret_access_key" in line:
            aws_key = re.sub("^.*aws_secret_access_key\s*=\s*", "", line)
conf.set("spark.hadoop.fs.s3a.access.key", aws_id)
conf.set("spark.hadoop.fs.s3a.secret.key", aws_key)
aws_id, aws_key = (None, None)
### end getting keys
sc = SparkContext(master = "spark://jupyter.corp.penoles.mx:7077",
                  sparkHome="/usr/local/spark/",
                  appName="examen-ma-2", conf=conf)
spark = SQLContext(sc)
## setup sparknlp source
##
## https://github.com/JohnSnowLabs/spark-nlp/issues/106
## https://stackoverflow.com/questions/34302314/no-module-name-pyspark-error
sys.path.extend(glob.glob("/usr/local/spark-2.3.0-bin-hadoop2.7/jars/spark-nlp_
2.11-1.5.3.jar"))
from sparknlp.annotator import *
from sparknlp.common import *
from sparknlp.base import *
from pyspark.ml import Pipeline
from pyspark.sql.functions import *
from pyspark.ml.fpm import FPGrowth
from pyspark.ml.feature import Word2Vec
import pyspark.sql.functions as sparkFunctions

train2 = spark.read.parquet("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/
```

```
train2.parquet").cache()
train2.show(1)
```

```
+-----+-------+--------------------+
|   id|cuisine|         ingredients|
+-----+-------+--------------------+
|10259|  greek|romaine lettuce, ...|
+-----+-------+--------------------+
only showing top 1 row
```

Custom Pipeline Transformer

In [2]:
```
# taken from https://blog.insightdatascience.com/spark-pipelines-elegant-yet-pow
erful-7be93afcdd42
# and https://stackoverflow.com/a/32337101/7323086
from pyspark import keyword_only
from pyspark.ml.pipeline import Transformer
from pyspark.ml.param.shared import HasInputCol, HasOutputCol
from pyspark.ml.feature import VectorAssembler, StringIndexer, IndexToString, ID
F, HashingTF, IDF, RegexTokenizer
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.classification import RandomForestClassifier
# Create a custom word count transformer class
class StringArray2PlainString(Transformer, HasInputCol, HasOutputCol):
    @keyword_only
    def __init__(self, inputCol=None, outputCol=None):
        super(StringArray2PlainString, self).__init__()
        kwargs = self._input_kwargs
        self.setParams(**kwargs)
    @keyword_only
    def setParams(self, inputCol=None, outputCol=None):
        kwargs = self._input_kwargs
        return self._set(**kwargs)
    def _transform(self, dataset):
        out_col = self.getOutputCol()
#         in_col = dataset[self.getInputCol()]
        in_col = self.getInputCol()
        finishr = Finisher() \
            .setInputCols([in_col]) \
            .setOutputCols(["tmp_col"]) \
            .setIncludeKeys(False) \
            .setAnnotationSplitSymbol(" ")
        tmp_ds = finishr.transform(dataset).cache()
        return tmp_ds.withColumn(out_col, sparkFunctions.split(tmp_ds.tmp_col,
",")).drop("tmp_col")
```

Pipeline

In [3]:
```
%%time
docAssemblr = DocumentAssembler()\
  .setInputCol("ingredients")\
  .setOutputCol("document")

tokenizr = Tokenizer() \
    .setInputCols(["document"]) \
    .setOutputCol("tokens")#    .addInfixPattern("(\p{L}+)(n't\b)") \

normalizr = Normalizer() \
    .setInputCols(["tokens"]) \
    .setOutputCol("normalized") \
    .setPattern("[^A-Za-z,]")

# path_dict = "file:/home/jaa6766/enwiki-latest-all-titles-in-ns0-transform"
path_dict = "hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/wiki-titles.txt"
norvig = NorvigSweetingApproach() \
    .setInputCols(["normalized"]) \
    .setOutputCol("ingredients2") \
    .setDictionary(path_dict)
# norvig.setCorpus("hdfs://jupyter.corp.penoles.mx:9000/spell-dicts/enwiki-lates
t-all-titles-in-ns0-transform")

stemmr2 = Stemmer() \
  .setInputCols(["ingredients2"]) \
  .setOutputCol("stems")

ar2str = StringArray2PlainString(
    inputCol="stems",
    outputCol="ingredients3")

word2v = Word2Vec() \
    .setVectorSize(40) \
    .setInputCol("ingredients3") \
    .setOutputCol("word2vec")

idf0 = IDF() \
    .setInputCol("ingredients3") \
    .setOutputCol("idf")

si0 = StringIndexer(inputCol="cuisine", outputCol="label")

va0 = VectorAssembler(inputCols=["idf"], outputCol="features")

pipeline = Pipeline(stages = [
    docAssemblr,
    tokenizr,
    normalizr,
    norvig,
    stemmr2,
    ar2str,
#     word2v,
#     idf0,
    si0,
#     va0
])
model_data = pipeline.fit(train2)
train4 = model_data.transform(train2).cache()
train4.show(2)
```

```
+-----+-----------+--------------------+--------------------+-----+
|   id|    cuisine|         ingredients|        ingredients3|label|
+-----+-----------+--------------------+--------------------+-----+
|10259|      greek|romaine lettuce, ...|[romain lettuc , ...|  9.0|
|25693|southern_us|plain flour, grou...|[plain flour ,  g...|  2.0|
+-----+-----------+--------------------+--------------------+-----+
only showing top 2 rows

CPU times: user 182 ms, sys: 41.4 ms, total: 223 ms
Wall time: 2min 47s
```

In [4]:
```
class ArrayString2String(Transformer, HasInputCol, HasOutputCol):
    @keyword_only
    def __init__(self, inputCol=None, outputCol=None):
        super(ArrayString2String, self).__init__()
        kwargs = self._input_kwargs
        self.setParams(**kwargs)
    @keyword_only
    def setParams(self, inputCol=None, outputCol=None):
        kwargs = self._input_kwargs
        return self._set(**kwargs)
    def _transform(self, dataset):
        out_col = self.getOutputCol()
        in_col = self.getInputCol()
        return (dataset
                .withColumn("tmp_col0", dataset[in_col].cast(StringType()))
                .withColumn("tmp_col1", regexp_replace("tmp_col0", "[\[\]]", ""
))
                .withColumn("tmp_col2", regexp_replace("tmp_col1", "\s+", " "))
                .withColumn(out_col, regexp_replace("tmp_col2", "\s*,\s*", ","))
                .drop("tmp_col0", "tmp_col1", "tmp_col2")
               )
```

In [5]:
```
###############
## IDF!!!
ars2s = ArrayString2String(inputCol="ingredients3", outputCol="ingredients4")
tknzr = RegexTokenizer(inputCol="ingredients4", outputCol="ingredients5", patter
n=",")
hshng = HashingTF(inputCol="ingredients5", outputCol="ingredients6")
idf0  = IDF(inputCol="ingredients6", outputCol="features")
pipe_hash = Pipeline(stages = [ars2s, tknzr, hshng, idf0])
pipe_hashm = pipe_hash.fit(train4.cache())
train5 = (pipe_hashm
 .transform(train4)
 .drop("ingredients", "ingredients3", "ingredients4", "ingredients5", "ingredien
ts6")
 .cache())
#  .show(truncate=False))
```

In [6]:
```
%%time
##############
## IDF!!!
(train_data, test_data) = train5.randomSplit([0.7, 0.3], seed = 175904)
ml0 = LogisticRegression(predictionCol="preds")
is0 = IndexToString(inputCol="preds",
                    labels=model_data.stages[6].labels,
                    outputCol="y_hat")
pipeml = Pipeline(stages=[ml0, is0])
model_ml = pipeml.fit(train_data)
test_data = model_ml.transform(test_data)
```

```
CPU times: user 151 ms, sys: 38.4 ms, total: 189 ms
Wall time: 3min 52s
```

In [7]:
```
ev0 = MulticlassClassificationEvaluator()
ev0.setLabelCol("label")
ev0.setPredictionCol("preds")
score0 = ev0.evaluate(test_data)
print("Score para Logistic Regression: %f (measure %s)"%(
    score0, ev0.getMetricName()
))
```

```
Score para Logistic Regression: 0.680478 (measure f1)
```

In [8]:
```
test_data.show(1)
```

```
+---+-------+-----+--------------------+--------------------+--------------------
-+-----+-------+
| id|cuisine|label|            features|       rawPrediction|         probabilit
y|preds|  y_hat|
+---+-------+-----+--------------------+--------------------+--------------------
-+-----+-------+
| 10|chinese|  4.0|(262144,[39249,79...|[-32.416457302380...|[1.7881308253111
9...| 14.0|british|
+---+-------+-----+--------------------+--------------------+--------------------
-+-----+-------+
only showing top 1 row
```

In [9]:
```
test_data.select("id", "cuisine", "y_hat").show(100)
```

id	cuisine	y_hat
10	chinese	british
1000	mexican	mexican
10003	french	spanish
10006	italian	italian
10009	japanese	japanese
10025	korean	korean
10029	japanese	spanish
10032	mexican	brazilian
10033	british	italian
10034	chinese	southern_us
10035	italian	italian
10037	korean	french
10038	french	italian
1004	vietnamese	vietnamese
10040	italian	italian
10042	southern_us	irish
10043	southern_us	mexican
10047	french	italian
10049	thai	thai
10055	chinese	chinese
1006	chinese	japanese
10062	southern_us	southern_us
10065	mexican	mexican
1007	greek	greek
10078	french	french
10082	italian	italian
10085	french	french
10086	italian	italian
10087	vietnamese	mexican
10090	southern_us	southern_us
10091	mexican	greek
1010	southern_us	irish
10108	indian	indian
10114	southern_us	southern_us
10120	indian	indian
10122	french	southern_us
10133	italian	italian
10139	italian	italian
10140	french	spanish
10152	mexican	mexican
10154	brazilian	mexican
10164	indian	chinese
10165	mexican	jamaican
10168	italian	italian
10175	mexican	mexican
10182	italian	southern_us
10188	indian	indian
10194	french	french
10203	french	french
10204	italian	italian
10215	thai	thai
10217	mexican	spanish
10219	greek	greek
10220	italian	italian
10223	italian	italian
10225	mexican	mexican
10226	chinese	chinese
10227	indian	japanese
10228	moroccan	moroccan
10239	mexican	mexican
10241	irish	irish

```
|10242|     mexican|     mexican|
|10247|     mexican|     mexican|
|10256|    japanese|     chinese|
|10258|      indian|      indian|
|10260| southern_us| southern_us|
|10264|      french|     italian|
|10265|    moroccan|    japanese|
|10275| southern_us| southern_us|
| 1028| southern_us| southern_us|
|10285|     mexican| southern_us|
|10293|     chinese|     chinese|
|10298|      indian|      indian|
|10303| southern_us| southern_us|
|10305|     mexican|     mexican|
| 1031|      indian|    japanese|
|10310|     italian|     italian|
|10312|     italian|     italian|
|10323|     italian|     italian|
|10324|     mexican|     mexican|
|10327|      french|      french|
| 1033|    moroccan|    moroccan|
|10335|     mexican|     mexican|
|10337|       greek|     spanish|
|10341|    moroccan|      french|
|10344| southern_us|     chinese|
|10347|      french|     mexican|
|10350|     italian|     italian|
|10351|     italian|     british|
| 1036|     chinese|     chinese|
|10364|     british| southern_us|
|10365|     chinese|     chinese|
|10371|     mexican|     mexican|
|10372|cajun_creole|cajun_creole|
|10377|     chinese|     chinese|
|10385|        thai|  vietnamese|
|10391|     italian|     italian|
|10394|      indian|      indian|
|10402|     chinese|     chinese|
|10409|cajun_creole|cajun_creole|
+-----+------------+------------+
only showing top 100 rows
```

Set de Pruebas Kaggle

In [10]:
```
test = None
try:
    print("Trying to read parquet...", end="")
    test = spark.read.parquet("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen
2/test.parquet")
    print(" OK!")
except:
    print(" Failed!!!\nReading from JSON...", end="")
    schema_ingredientes_test = schema=StructType().\
        add("id", data_type=StringType(), nullable=False, metadata=None).\
        add("ingredients", data_type=ArrayType(StringType()), nullable=True, met
adata=None)
    test = spark.read.json("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/t
est.json",
                           schema=schema_ingredientes_test,
                           allowUnquotedFieldNames=True,
                           multiLine=True)
    test = test \
        .withColumn("ingreds",
                    col("ingredients").cast(StringType())) \
        .withColumn("ingredientes",
                   regexp_replace(col("ingreds"), pattern="[\[\]]", replacement=
""))\
        .select("id", col("ingredientes").alias("ingredients"))
    test.write.parquet("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/test.
parquet", mode="overwrite")
    test = spark.read.parquet("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen
2/test.parquet")
    print(" Done")
```

```
Trying to read parquet... OK!
```

Pipeline de adecuación de datos

In [11]:
```
##############
## IDF!!!
test2 = pipe_hashm.transform(model_data.transform(test)).select("id", "features"
).cache()
test2.show()
```

```
+-----+--------------------+
|   id|            features|
+-----+--------------------+
|18009|(262144,[45688,10...|
|28583|(262144,[46588,52...|
|41580|(262144,[80021,99...|
|29752|(262144,[1176,263...|
|35687|(262144,[30649,39...|
|38527|(262144,[53031,63...|
|19666|(262144,[174095,2...|
|41217|(262144,[6113,439...|
|28753|(262144,[9790,207...|
|22659|(262144,[6767,130...|
|21749|(262144,[6113,726...|
|44967|(262144,[9879,141...|
|42969|(262144,[108254,1...|
|44883|(262144,[4065,296...|
|20827|(262144,[94567,10...|
|23196|(262144,[21625,55...|
|35387|(262144,[7977,135...|
|33780|(262144,[3419,204...|
|19001|(262144,[37601,63...|
|16526|(262144,[17468,43...|
+-----+--------------------+
only showing top 20 rows
```

Pipeline ML

In [13]:
```
%%time
test_out = model_ml.transform(test2)
test_out2 = test_out.select("id", test_out.y_hat.alias("cuisine"))
test_out2.show()
```

```
+-----+------------+
|   id|     cuisine|
+-----+------------+
|18009|     british|
|28583| southern_us|
|41580|     italian|
|29752|cajun_creole|
|35687|     italian|
|38527| southern_us|
|19666| southern_us|
|41217|     chinese|
|28753|     mexican|
|22659|     russian|
|21749|     italian|
|44967|       greek|
|42969|      indian|
|44883|     italian|
|20827|     british|
|23196|     italian|
|35387|     mexican|
|33780|     mexican|
|19001|     mexican|
|16526|      korean|
+-----+------------+
only showing top 20 rows

CPU times: user 39 ms, sys: 14.2 ms, total: 53.1 ms
Wall time: 366 ms
```

In [14]:
```
(test_out2
 .coalesce(1)
 .write
 .csv("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/test_submit.csv",
     header=True,
     mode="overwrite"))
```

Pruebas

In [15]:
```
def test_star(*args, **kwargs):
    return ([x for x in args],
            kwargs)

test_star('a', 'b', c='c', d='d')
```

Out[15]:
```
(['a', 'b'], {'c': 'c', 'd': 'd'})
```

/Prueba

Fin del Cluster

In [16]:
```
sc.stop()
```

Bibliografía

- Notas del Curso Métodos Analíticos, Luis Felipe González, ITAM Primavera 2018 (https://clever-mestorf-ee3f54.netlify.com)
- https://github.com/JohnSnowLabs/spark-nlp/blob/master/python/example/model-downloader/ModelDownloaderExample.ipynb (https://github.com/JohnSnowLabs/spark-nlp/blob/master/python/example/model-downloader/ModelDownloaderExample.ipynb)
- https://nlp.johnsnowlabs.com/components.html (https://nlp.johnsnowlabs.com/components.html)
- https://nlp.johnsnowlabs.com/notebooks.html (https://nlp.johnsnowlabs.com/notebooks.html)
- https://github.com/JohnSnowLabs/spark-nlp/blob/1.5.0/python/example/vivekn-sentiment/sentiment.ipynb (https://github.com/JohnSnowLabs/spark-nlp/blob/1.5.0/python/example/vivekn-sentiment/sentiment.ipynb)
- Indix - Lessons from Using Spark to Process Large Amounts of Data – Part I. Retrieved 2018-05-14 (https://www.indix.com/blog/engineering/lessons-from-using-spark-to-process-large-amounts-of-data-part-i/)
- Spark NLP - Dependencies (https://mvnrepository.com/artifact/com.johnsnowlabs.nlp/spark-nlp_2.11/1.5.3)
- StackOverflow: Troubleshotting on Spark (https://stackoverflow.com/a/36903019/7323086)
- https://github.com/JohnSnowLabs/spark-nlp/issues/106 (https://github.com/JohnSnowLabs/spark-nlp/issues/106)
- https://stackoverflow.com/questions/34302314/no-module-name-pyspark-error (https://stackoverflow.com/questions/34302314/no-module-name-pyspark-error)