Examen 2

175904 - Jorge III Altamirano Astorga

176576 - Rodrigo Cedeño

177508 - Uriel Miranda Miñón

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
```

In [2]:
```
# 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.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)
conf.set("spark.jars.packages", "JohnSnowLabs:spark-nlp:1.5.3")
sc = SparkContext(master = "spark://jupyter.corp.penoles.mx:7077",
                  sparkHome="/usr/local/spark/",
                  appName="examen-ma-2", conf=conf)
spark = SQLContext(sc)
sc.version
```

Out[2]:
```
'2.3.0'
```

In [3]:
```
%%time
schema_ingredientes = schema=StructType().\
    add("id", data_type=StringType(), nullable=False, metadata=None).\
    add("cuisine", data_type=StringType(), nullable=False, metadata=None).\
    add("ingredients", data_type=ArrayType(StringType()), nullable=True, metadat
a=None)
train = spark.read.json("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/trai
n.json",
                        schema=schema_ingredientes,
                        allowUnquotedFieldNames=True,
                        multiLine=True)
print("Schema:")
train.printSchema()
print("Show:")
train.show(5)
```

```
Schema:
root
 |-- id: string (nullable = true)
 |-- cuisine: string (nullable = true)
 |-- ingredients: array (nullable = true)
 |    |-- element: string (containsNull = true)

Show:
+-----+-----------+--------------------+
|   id|    cuisine|         ingredients|
+-----+-----------+--------------------+
|10259|      greek|[romaine lettuce,...|
|25693|southern_us|[plain flour, gro...|
|20130|   filipino|[eggs, pepper, sa...|
|22213|     indian|[water, vegetable...|
|13162|     indian|[black pepper, sh...|
+-----+-----------+--------------------+
only showing top 5 rows

CPU times: user 9.64 ms, sys: 9.4 ms, total: 19 ms
Wall time: 9.31 s
```

Conteo de Registros

In [4]:
```
train.count()
```

Out[4]:
```
39774
```

Manipulación de la Columna

Quitamos los arrays, para operar mejor el machine learning.

In [5]:
```
train2 = train\
    .withColumn("ingreds",
                col("ingredients").cast(StringType()))\
    .withColumn("ingredientes",
               regexp_replace(col("ingreds"), pattern="[\[\]]", replacement=""))
\
    .select("id", "cuisine", col("ingredientes").alias("ingredients"))
train2.write.parquet("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/train2.
parquet", mode="overwrite")
```

In [6]:
```
train2 = spark.read.parquet("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/
train2.parquet")
train2.show()
```

```
+-----+-----------+--------------------+
|   id|    cuisine|         ingredients|
+-----+-----------+--------------------+
|10259|      greek|romaine lettuce, ...|
|25693|southern_us|plain flour, grou...|
|20130|   filipino|eggs, pepper, sal...|
|22213|     indian|water, vegetable ...|
|13162|     indian|black pepper, sha...|
| 6602|   jamaican|plain flour, suga...|
|42779|    spanish|olive oil, salt, ...|
| 3735|    italian|sugar, pistachio ...|
|16903|    mexican|olive oil, purple...|
|12734|    italian|chopped tomatoes,...|
| 5875|    italian|pimentos, sweet p...|
|45887|    chinese|low sodium soy sa...|
| 2698|    italian|Italian parsley l...|
|41995|    mexican|ground cinnamon, ...|
|31908|    italian|fresh parmesan ch...|
|24717|     indian|tumeric, vegetabl...|
|34466|    british|greek yogurt, lem...|
| 1420|    italian|italian seasoning...|
| 2941|       thai|sugar, hot chili,...|
| 8152| vietnamese|soy sauce, vegeta...|
+-----+-----------+--------------------+
only showing top 20 rows
```

Procesamiento de Lenguaje Natural

Carga de librerías Spark NLP

- 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)

In [7]:
```
## setup sparknlp source
##
## https://github.com/JohnSnowLabs/spark-nlp/issues/106
## https://stackoverflow.com/questions/34302314/no-module-name-pyspark-error
import os, glob, sys
sys.path.extend(glob.glob("/usr/local/spark-2.3.0-bin-hadoop2.7/jars/spark-nlp_
2.11-1.5.0.jar"))
from sparknlp.annotator import *
from sparknlp.common import *
from sparknlp.base import *
from pyspark.ml import Pipeline
import os, glob, sys
from pyspark.sql.functions import *
from pyspark.ml.fpm import FPGrowth
```

Pipeline

In [8]:

```
%%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,]")

stemmr = Stemmer() \
  .setInputCols(["normalized"]) \
  .setOutputCol("stems")

finishr = Finisher() \
    .setInputCols(["stems"]) \
    .setOutputCols(["ingredients"]) \
    .setIncludeKeys(False)

pipeline = Pipeline(stages = [
    docAssemblr,
    tokenizr,
    normalizr,
    stemmr,
    finishr
])

train.cache()
model = pipeline.fit(train2)
train3 = model.transform(train2)
train3.printSchema()
print("showing results...")
train3.show(2, truncate=False)
```

```
root
 |-- id: string (nullable = true)
 |-- cuisine: string (nullable = true)
 |-- ingredients: string (nullable = true)

showing results...
+-----+-----------+------------------------------------------------------------
-------------------------------------------------------------------+
|id   |cuisine    |ingredients
                                                                   |
+-----+-----------+------------------------------------------------------------
-------------------------------------------------------------------+
|10259|greek      |romain@lettuc@,@black@oliv@,@grape@tomato@,@garlic@,@pepper@,
@purpl@onion@,@season@,@garbanzo@bean@,@feta@chees@crumbl          |
|25693|southern_us|plain@flour@,@ground@pepper@,@salt@,@tomato@,@ground@black@pe
pper@,@thym@,@egg@,@green@tomato@,@yellow@corn@meal@,@milk@,@veget@oil|
+-----+-----------+------------------------------------------------------------
-------------------------------------------------------------------+
only showing top 2 rows

CPU times: user 63.7 ms, sys: 10.5 ms, total: 74.2 ms
Wall time: 1.55 s
```

Canastas

En este primer paso lo que realizamos es volver a pasar los datos a array, para que sean canastas

In [9]:
```
%%time
udf_ingredients = udf(lambda ingredients:
                      list(set(ingredients)),
                      returnType=ArrayType(StringType()))
train4 = train3 \
    .withColumn("ingredients", regexp_replace("ingredients", "@?,@?", ",")) \
    .select("id", "cuisine",
        split("ingredients", "\s*,\s*").alias("ingredients")) \
    .cache() \
    .withColumn("ingredients", udf_ingredients("ingredients"))
#.select( \
#        "id",
#        "cuisine",
#        regexp_replace("ingredients", "\@", "@ ").alias("ingredients")\
#    ) \
train4.write.parquet("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/tmp-tra
in4.parquet", mode="overwrite")
```

```
CPU times: user 33.6 ms, sys: 17 ms, total: 50.6 ms
Wall time: 41.4 s
```

In [10]:
```
train4 = spark.read.parquet("hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/
tmp-train4.parquet")
train4.printSchema()
train4.show()
```

```
root
 |-- id: string (nullable = true)
 |-- cuisine: string (nullable = true)
 |-- ingredients: array (nullable = true)
 |    |-- element: string (containsNull = true)

+-----+-----------+--------------------+
|   id|    cuisine|         ingredients|
+-----+-----------+--------------------+
|10259|      greek|[purpl@onion, bla...|
|25693|southern_us|[veget@oil, salt,...|
|20130|   filipino|[chicken@liver, g...|
|22213|     indian|[water, veget@oil...|
|13162|     indian|[water, cayenn@pe...|
| 6602|   jamaican|[ground@ginger, f...|
|42779|    spanish|[flat@leaf@parsle...|
| 3735|    italian|[flour, white@alm...|
|16903|    mexican|[iceberg@lettuc, ...|
|12734|    italian|[flat@leaf@parsle...|
| 5875|    italian|[mushroom, canola...|
|45887|    chinese|[crush@red@pepper...|
| 2698|    italian|[italian@parslei@...|
|41995|    mexican|[avocado, crush@r...|
|31908|    italian|[allpurpos@flour,...|
|24717|     indian|[spinach, sweet@p...|
|34466|    british|[confection@sugar...|
| 1420|    italian|[italian@season, ...|
| 2941|       thai|[asian@fish@sauc,...|
| 8152| vietnamese|[veget@oil, chick...|
+-----+-----------+--------------------+
only showing top 20 rows
```

Items frecuentes

In [11]:
```
%%time
fp = FPGrowth(minSupport=0.1, minConfidence=0.2, itemsCol="ingredients")
fpm = fp.fit(train4)
```

```
CPU times: user 17.5 ms, sys: 1.62 ms, total: 19.1 ms
Wall time: 6.56 s
```

In [12]:
```
fpm.freqItemsets.orderBy(col("freq").desc()).show(truncate=False)
```

```
+---------------------+-----+
|items                |freq |
+---------------------+-----+
|[salt]               |18048|
|[onion]              |7972 |
|[oliv@oil]           |7971 |
|[water]              |7457 |
|[garlic]             |7380 |
|[sugar]              |6434 |
|[garlic@clove]       |6236 |
|[butter]             |4847 |
|[ground@black@pepper]|4784 |
|[allpurpos@flour]    |4632 |
|[pepper]             |4438 |
|[onion, salt]        |4392 |
|[veget@oil]          |4385 |
|[oliv@oil, salt]     |4177 |
+---------------------+-----+
```

Reglas de Asociación y Predicciones

Al parecer no se observan reglas de asociación. Por ende, no hay predicciones

In [13]:
```
fpm.associationRules.show()
```

```
+----------+----------+-------------------+
|antecedent|consequent|         confidence|
+----------+----------+-------------------+
|    [salt]|[oliv@oil]|0.23143838652482268|
|    [salt]|   [onion]|0.24335106382978725|
|[oliv@oil]|    [salt]| 0.5240245891356166|
|   [onion]|    [salt]| 0.5509282488710486|
+----------+----------+-------------------+
```

In [14]:
```
fpm.transform(train4).show(10, truncate=False)
```

```
+-----+-----------+---------------------------------------------------------------
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
-------------+----------------+
|id   |cuisine    |ingredients


             |prediction      |
+-----+-----------+---------------------------------------------------------------
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
-------------+----------------+
|10259|greek      |[purpl@onion, black@oliv, season, romain@lettuc, pepper, garl
ic, feta@chees@crumbl, grape@tomato, garbanzo@bean]

             |[]              |
|25693|southern_us|[veget@oil, salt, green@tomato, milk, tomato, thym, egg, yell
ow@corn@meal, ground@pepper, ground@black@pepper, plain@flour]

             |[oliv@oil, onion]|
|20130|filipino   |[chicken@liver, grill@chicken@breast, mayonais, salt, yellow@
onion, cook@oil, pepper, egg, garlic@powder, soi@sauc, green@chili, butter]

             |[oliv@oil, onion]|
|22213|indian     |[water, veget@oil, salt, wheat]


             |[oliv@oil, onion]|
|13162|indian     |[water, cayenn@pepper, doubl@cream, black@pepper, boneless@ch
icken@skinless@thigh, lemon@juic, natur@yogurt, ground@cumin, chili@powder, garl
ic@past, milk, salt, bai@leaf, oil, passata, cornflour, shallot, butter, onion,
garam@masala]|[oliv@oil]       |
|6602 |jamaican   |[ground@ginger, fresh@ginger@root, sugar, salt, milk, egg, va
nilla@extract, powder@sugar, plain@flour, butter, bake@powder, ground@cinnamon]

             |[oliv@oil, onion]|
|42779|spanish    |[flat@leaf@parslei, skirt@steak, medium@shrimp, oliv@oil, whi
te@vinegar, salt, bai@leaf, pepper, garlic, chop@cilantro, jalapeno@chili, sea@s
alt, chorizo@sausag]
             |[onion]         |
|3735 |italian    |[flour, white@almond@bark, almond@extract, pistachio@nut, sug
ar, oliv@oil, egg, vanilla@extract, dri@cranberri, bake@powder]

             |[salt]          |
|16903|mexican    |[iceberg@lettuc, cheddar@chees, purpl@onion, oliv@oil, lime,
salt, fresh@pineappl, corn@tortilla, jalapeno@chili, poblano@pepper, ground@blac
k@pepper, pork, chop@cilantro@fresh]
             |[onion]         |
|12734|italian    |[flat@leaf@parslei, kosher@salt, fresh@basil, extravirgin@oli
v@oil, garlic, chop@tomato]

             |[]              |
+-----+-----------+---------------------------------------------------------------
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
-------------+----------------+
only showing top 10 rows
```

Ejemplos con ingredientes arbitrarios

- **Salt**, Eggs

In [15]:
```
fpm.transform(spark.createDataFrame([(["salt", "eggs"], )], ["ingredients"])).sh
ow()
```

```
+------------+-----------------+
| ingredients|       prediction|
+------------+-----------------+
|[salt, eggs]|[oliv@oil, onion]|
+------------+-----------------+
```

In [16]:
```
spark.registerDataFrameAsTable(train3, "train3")
spark.registerDataFrameAsTable(train4, "train4")
ingredients_nonunique = spark\
    .sql("SELECT ingredients FROM train4 WHERE id = 1667 OR \
         array_contains(ingredients, 'old@ el@ paso@ mild@ red@ enchilada@ sauc@
 ')") \
    .collect()[0].ingredients
ingredients_nonunique
```

Out[16]:
```
['old@el@paso@mild@red@enchilada@sauc',
 'cook@chicken',
 'mexican@chees@blend',
 'pillsburi@refriger@crescent@dinner@roll',
 'red@enchilada@sauc',
 'refriger@crescent@roll']
```

- Lettuce, Tomato, **Olive@oil**

In [17]:
```
fpm.transform(spark.createDataFrame([(["lettuce", "tomato", "oliv@oil"], )], ["i
ngredients"])).show(truncate=False)
```

```
+---------------------------+----------+
|ingredients                |prediction|
+---------------------------+----------+
|[lettuce, tomato, oliv@oil]|[salt]    |
+---------------------------+----------+
```

In [18]:
```
%%time
train4.select(explode("ingredients")).distinct().count()
```

```
CPU times: user 5.59 ms, sys: 4.83 ms, total: 10.4 ms
Wall time: 2.65 s
```

Out[18]:
```
6681
```

Títulos de Wikipedia

In [19]:
```
!cat wikipedia-title-corpus-download.sh
```

```
#!/bin/sh
#english español Deutsch portugués francais tagalog italiano vietnamese
source ~/.bash_profile
for lang in en es de pt fr lt it vi;
do
        echo "Downloading $lang"
        wget -c "https://dumps.wikimedia.org/${lang}wiki/latest/${lang}wiki-late
st-all-titles-in-ns0.gz"
        echo -n Transforming $lang...
        zcat ${lang}wiki-latest-all-titles-in-ns0.gz | \
                 sed 's!_\+! !g;s![^a-z ]!!ig;s!^\s\+!!;s!\s\+$!!;/^\s*$/d' | \
                 tr '[:upper:]' '[:lower:]' | tr ' ' '\n' | sort -u > "${lang}wi
ki-latest-all-titles-in-ns0-transform"
        echo " done!"
done
cat *-transform > wiki-titles.txt
hdfs dfs -copyFromLocal wiki-titles.txt hdfs://jupyter.corp.penoles.mx:9000/ma20
18-examen2/
```

In [20]:
```
%%time
stemmr2 = Stemmer() \
  .setInputCols(["tokens"]) \
  .setOutputCol("stems")
path_dict = "hdfs://jupyter.corp.penoles.mx:9000/ma2018-examen2/wiki-titles.txt"
norvig = NorvigSweetingApproach()
norvig.setInputCols(["stems"])
norvig.setOutputCol("ingredients2")
norvig.setDictionary(path_dict)
finishr2 = Finisher() \
    .setInputCols(["ingredients2"]) \
    .setOutputCols(["ingredients3"]) \
    .setIncludeKeys(False) \
    .setAnnotationSplitSymbol(" ")


pipeline1 = Pipeline(stages = [
    docAssemblr,
    tokenizr,
    stemmr2,
    norvig,
    finishr2
])
model1 = pipeline1.fit(train2)
train4 = model1.transform(train2)
train4.show(2)
```

```
+-----+-----------+--------------------+--------------------+
|   id|    cuisine|         ingredients|        ingredients3|
+-----+-----------+--------------------+--------------------+
|10259|      greek|romaine lettuce, ...|romain lettuc , b...|
|25693|southern_us|plain flour, grou...|plain flour , gro...|
+-----+-----------+--------------------+--------------------+
only showing top 2 rows

CPU times: user 102 ms, sys: 26.7 ms, total: 128 ms
Wall time: 2min 13s
```

Word2Vec

In [21]:
```
from pyspark.ml.feature import Word2Vec
import pyspark.sql.functions as sparkFunctions
```

In [22]:
```
%%time
train5 = train4.withColumn("ingredients2", sparkFunctions.split("ingredients3",
 ",")) \
         .cache()
word2v = Word2Vec(vectorSize=100, seed=175904, inputCol="ingredients2", outputCo
l="word2vec")
word2vm = word2v.fit(train5)
train_w2v = word2vm.transform(train5)
```

```
CPU times: user 62.9 ms, sys: 9.29 ms, total: 72.2 ms
Wall time: 1min 23s
```

In [23]:
```
train_w2v.show()
train_w2v.coalesce(1).write.json("s3a://jorge-altamirano/ma2018-examen2/word2vec
-small.json",
                                 compression="gzip",
                               mode="overwrite")
```

```
+-----+-----------+--------------------+--------------------+-------------------
-+--------------------+
|   id|    cuisine|         ingredients|        ingredients3|        ingredients
2|            word2vec|
+-----+-----------+--------------------+--------------------+-------------------
-+--------------------+
|10259|      greek|romaine lettuce, ...|romain lettuc , b...|[romain lettuc ,
...|[-0.0034980849983...|
|25693|southern_us|plain flour, grou...|plain flour , gro...|[plain flour ,
g...|[0.07107551432934...|
|20130|   filipino|eggs, pepper, sal...|egg , pepper , sa...|[egg ,  pepper ,
...|[0.03438723728080...|
|22213|     indian|water, vegetable ...|water , veget oil...|[water ,  veget
o...|[0.07553127699065...|
|13162|     indian|black pepper, sha...|black pepper , sh...|[black pepper ,
...|[0.07961993085300...|
| 6602|   jamaican|plain flour, suga...|plain flour , sug...|[plain flour ,
s...|[0.12873898710434...|
|42779|    spanish|olive oil, salt, ...|oliv oil , salt ,...|[oliv oil ,  sal
t...|[-0.0062195731756...|
| 3735|    italian|sugar, pistachio ...|sugar , pistachio...|[sugar ,  pistac
h...|[0.03430244047194...|
|16903|    mexican|olive oil, purple...|oliv oil , purpl ...|[oliv oil ,  pur
p...|[-0.0089989627818...|
|12734|    italian|chopped tomatoes,...|chop tomato , fre...|[chop tomato ,
f...|[0.02471002637563...|
| 5875|    italian|pimentos, sweet p...|pimento , sweet p...|[pimento ,  swee
t...|[-0.0240682438911...|
|45887|    chinese|low sodium soy sa...|low sodium soi sa...|[low sodium soi
s...|[-0.0647938878585...|
| 2698|    italian|Italian parsley l...|italian parslei l...|[italian parslei
...|[0.03446181102190...|
|41995|    mexican|ground cinnamon, ...|ground cinnamon ,...|[ground cinnamon
...|[0.01670660887954...|
|31908|    italian|fresh parmesan ch...|fresh parmesan ch...|[fresh parmesan
c...|[0.05624215947076...|
|24717|     indian|tumeric, vegetabl...|tumer , veget sto...|[tumer ,  veget
s...|[0.07939426787197...|
|34466|    british|greek yogurt, lem...|greek yogurt , le...|[greek yogurt ,
...|[0.03638339189637...|
| 1420|    italian|italian seasoning...|italian season , ...|[italian season
,...|[-0.0199193670996...|
| 2941|       thai|sugar, hot chili,...|sugar , hot chili...|[sugar ,  hot ch
i...|[-0.0345859354129...|
| 8152| vietnamese|soy sauce, vegeta...|soi sauc , veget ...|[soi sauc ,  veg
e...|[-0.0360192402552...|
+-----+-----------+--------------------+--------------------+-------------------
-+--------------------+
only showing top 20 rows
```

In [24]:
```
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)