"""

Walter Stevens

Using TF-IDF on Spark using Python

The example is to create a simple search engine for Wikipedia

1/9/2019

"""

```
from pyspark import SparkConf, SparkContext
from pyspark.mllib.feature import HashingTF
from pyspark.mllib.feature import IDF

# Standard Spark stuff:
conf = SparkConf().setMaster("local").setAppName("SparkTFIDF")
sc = SparkContext(conf = conf)

# Load documents (one per line).
rawData = sc.textFile("subset-small.tsv")
fields = rawData.map(lambda x: x.split("\t"))
documents = fields.map(lambda x: x[3].split(" "))

#[3] is the body of the text

# Store the document names for later:
documentNames = fields.map(lambda x: x[1])
#[1] is the document name
```

```
# Now hash the words in each document to their term frequencies:
hashingTF = HashingTF(100000)  #100K hash buckets just to save some memory
tf = hashingTF.transform(documents)

# At this point we have an RDD of sparse vectors representing each document,
# where each value maps to the term frequency of each unique hash value.

# Let's compute the TF*IDF of each term in each document:
tf.cache()
#because we will use it several times
idf = IDF(minDocFreq=2).fit(tf)
tfidf = idf.transform(tf)

# Now we have an RDD of sparse vectors, where each value is the TFxIDF
# of each unique hash value for each document.

# I happen to know that the article for "Abraham Lincoln" is in the data
# set, so let's search for "Gettysburg" (Lincoln gave a famous speech there):

# First, we figure out what hash value "Gettysburg" maps to by finding the
# index a sparse vector from HashingTF gives us back:
gettysburgTF = hashingTF.transform(["Gettysburg"])
gettysburgHashValue = int(gettysburgTF.indices[0])

# Now we will extract the TF*IDF score for Gettsyburg's hash value into
# a new RDD for each document:
gettysburgRelevance = tfidf.map(lambda x: x[gettysburgHashValue])

# We'll zip in the document names so we can see which is which:
zippedResults = gettysburgRelevance.zip(documentNames)
```

```
# And, print the document with the maximum TF*IDF value:
print("Best document for Gettysburg is:")
print(zippedResults.max())
```