



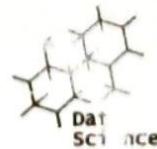
BIG DATA IN MACHINE LEARNING

Bài 10: PySpark Streaming

Phòng LT & Mạng

<https://cxt.edu.vn/vlap/tiep-tuc/Big-Data-In-Machine-Learning-198>

2020

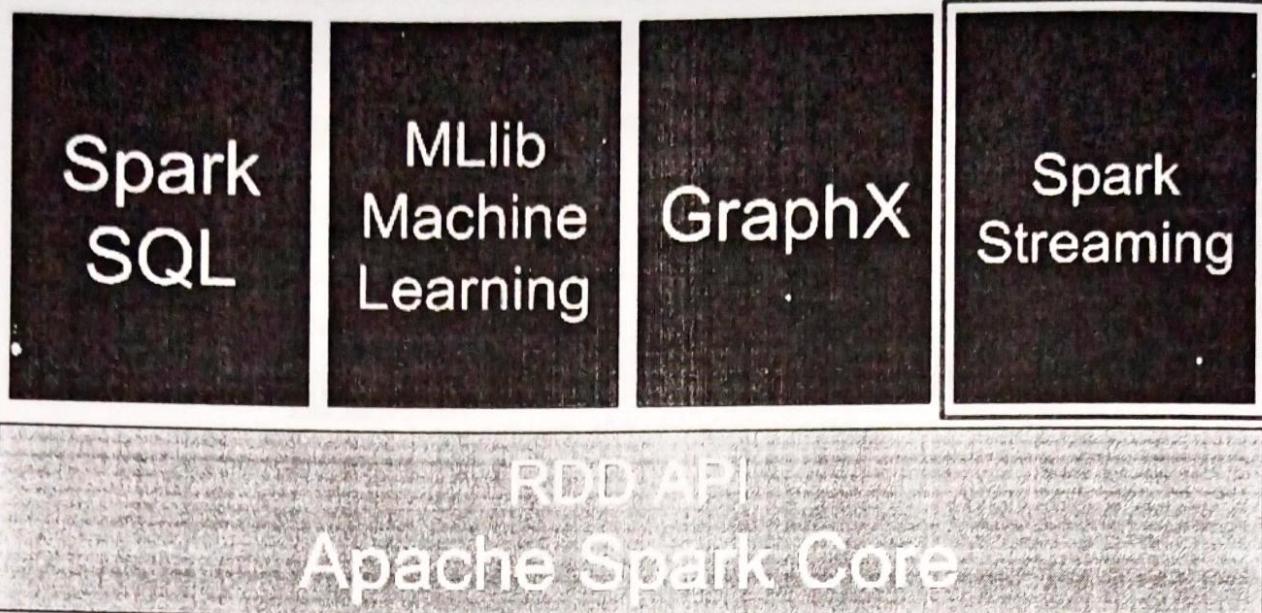


Nội dung



- Giới thiệu Spark Streaming
- Spark Streaming Fundamentals
- Làm việc với PySpark Streaming

Giới thiệu Spark Streaming



Giới thiệu Spark Streaming

- “Spark Streaming makes it easy to build scalable fault-tolerant streaming applications.”



Giới thiệu Spark Streaming

□ Spark Streaming

- Là một phần mở rộng của core API Spark cho phép xử lý luồng dữ liệu trực tiếp (live data stream), nó có khả năng mở rộng, thông lượng cao (high-throughput), có khả năng chịu lỗi (fault-tolerant).
- Có thể được sử dụng để truyền dữ liệu trực tiếp và xử lý có thể xảy ra trong thời gian thực.
- Số lượng user ngày càng cao, ví dụ như các thành viên trong Uber, Netflix và Pinterest...



Giới thiệu Spark Streaming

□ Data Streaming

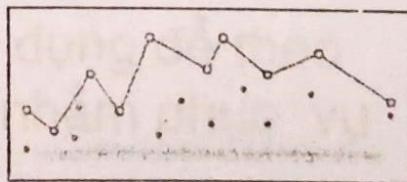
- Là một kỹ thuật truyền dữ liệu để dữ liệu có thể được xử lý như một luồng ổn định và liên tục.
- Với sự phát triển của Internet, các công nghệ Streaming ngày càng trở nên quan trọng.



User



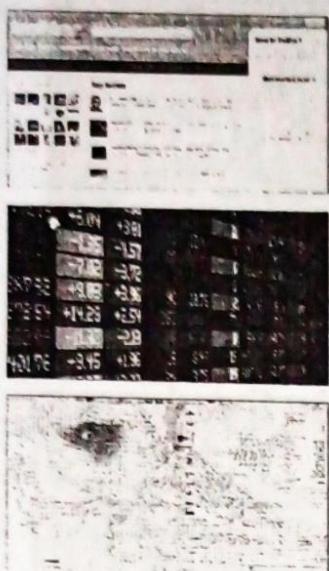
"Without stream processing there's no big data and no Internet of Things" – Dana Sandu, SQLstream



Live Stream Data

Giới thiệu Spark Streaming

☐ Tại sao chọn Spark Streaming?

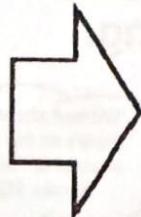


Spark Streaming is used to stream real-time data from various sources like Twitter, Stock Market and Geographical Systems and perform powerful analytics to help businesses.

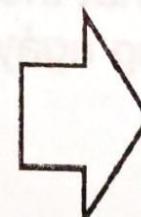


Giới thiệu Spark Streaming

Kafka
Flume
HDFS/S3
Kinesis
Twitter



Spark
Streaming



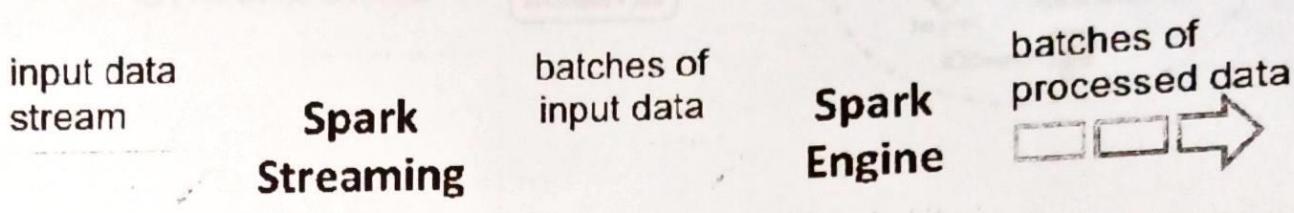
HDFS
Databases
Dashboards



Giới thiệu Spark Streaming

□ Streaming

- Trong nội tại của Spark Streaming, nó nhận các luồng dữ liệu đầu vào theo thời gian thực và chia dữ liệu thành các batch, sau đó được xử lý bởi Spark engine để tạo ra luồng kết quả cuối cùng theo từng batch.



Big Data in Machine Learning

9



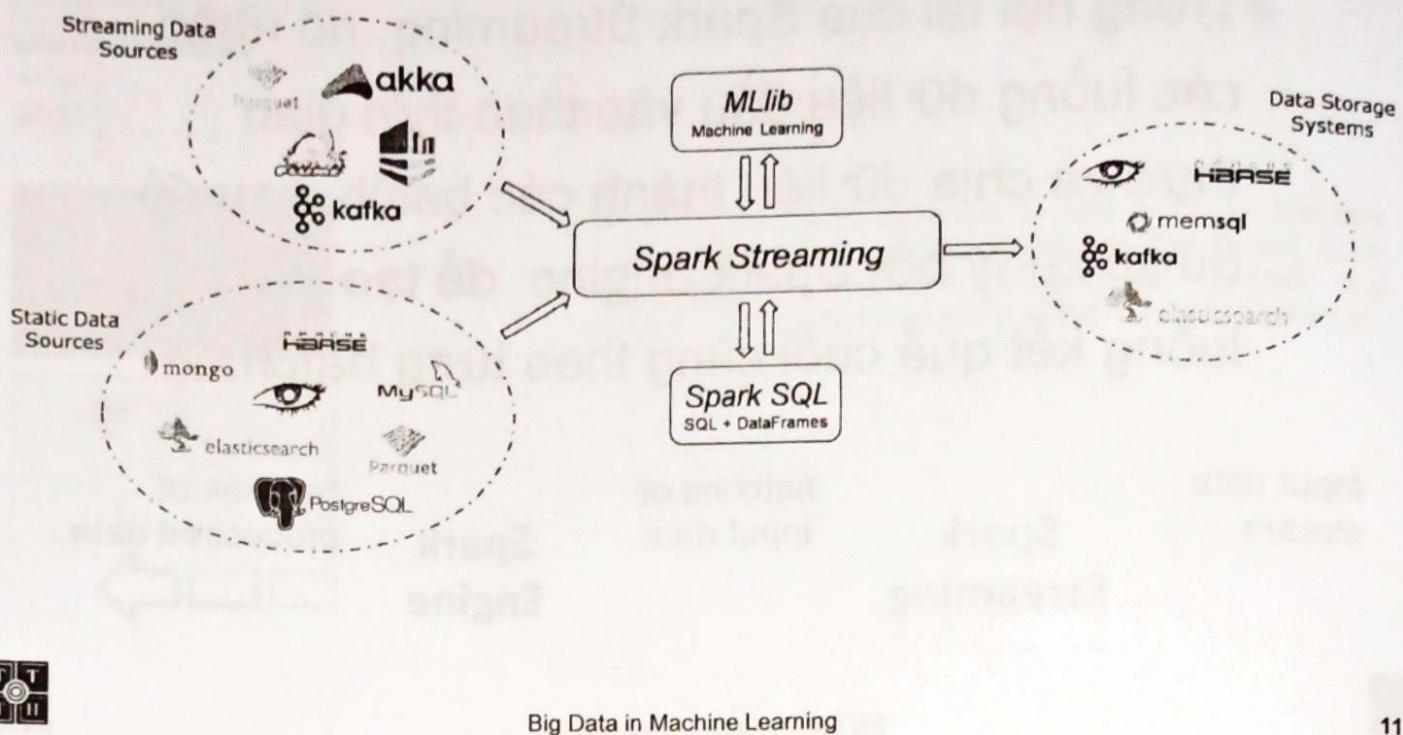
Giới thiệu Spark Streaming

□ Đặc điểm

- Scaling: có thể dễ dàng mở rộng đến hàng trăm node.
- Speed: đạt được độ trễ thấp (low latency).
- Fault Tolerance: có khả năng phục hồi hiệu quả từ các lỗi.
- Integration: tích hợp với batch & real-time processing.
- Business Analysis: được sử dụng để theo dõi hành vi của khách hàng nhằm phục vụ cho việc business analysis.

Giới thiệu Spark Streaming

Spark Streaming Workflow



Nội dung

1. Giới thiệu Spark Streaming
2. Spark Streaming Fundamentals
3. Làm việc với PySpark Streaming



Spark Streaming Fundamentals



❑ Gồm có

- Streaming Context
- DStream
- Caching
- Accumulators, Broadcast Variables & Checkpoints

Spark Streaming Fundamentals



❑ Streaming Context

- Streaming Context xử lý một luồng dữ liệu trong Spark. Nó đăng ký Input DStream để tạo Receiver object. Đây là main entry point cho Spark functionality.

Spark Streaming Fundamentals

- Spark cung cấp một số triển khai mặc định của các nguồn như Twitter, Akka Actor & ZeroMQ, có thể truy cập được từ context.



Figure: Spark Streaming Context



Figure: Default Implementation Sources



Spark Streaming Fundamentals

- Một StreamingContext object có thể được tạo từ SparkContext object. Một SparkContext đại diện cho kết nối đến Spark cluster và có thể được sử dụng để tạo RDD, accumulator và broadcast variable trên cluster đó.



□ Discretized Stream (DStream)

- Được cung cấp bởi Spark Streaming.
- Là một luồng dữ liệu liên tục
- Nhận từ nguồn dữ liệu (data source) hoặc luồng dữ liệu đã xử lý (processed data stream) được tạo ra bằng cách chuyển đổi input stream.

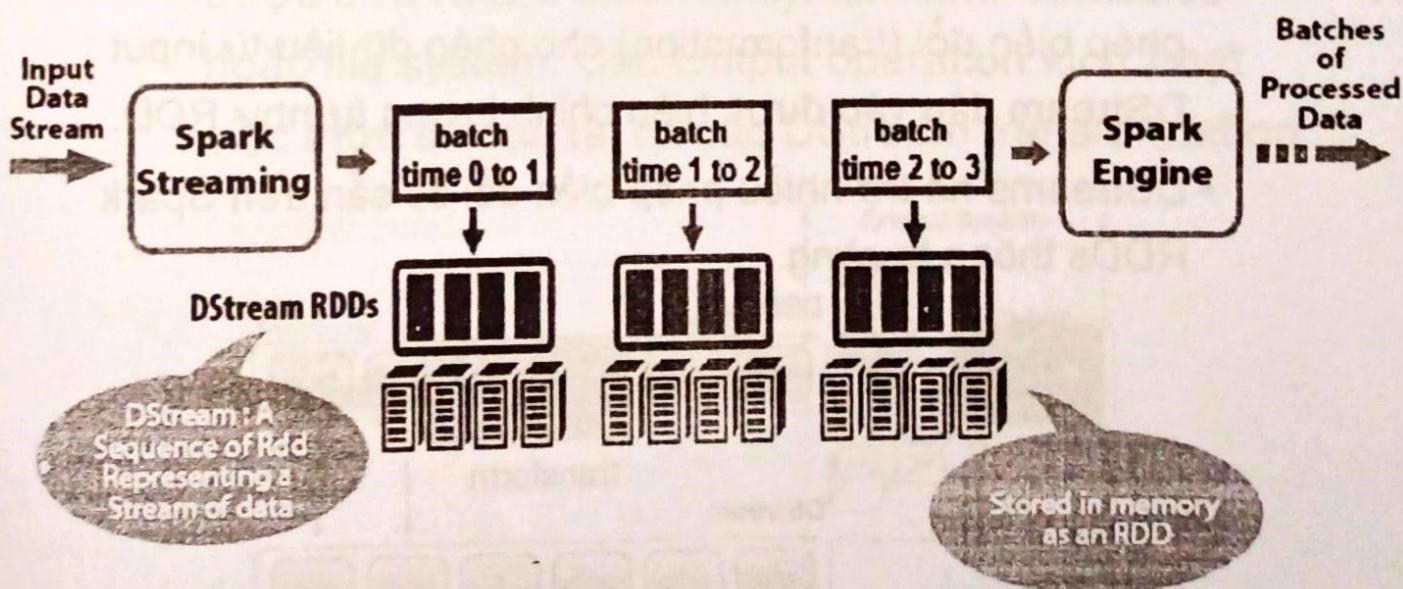
Big Data in Machine Learning

17

Spark Streaming Fundamentals



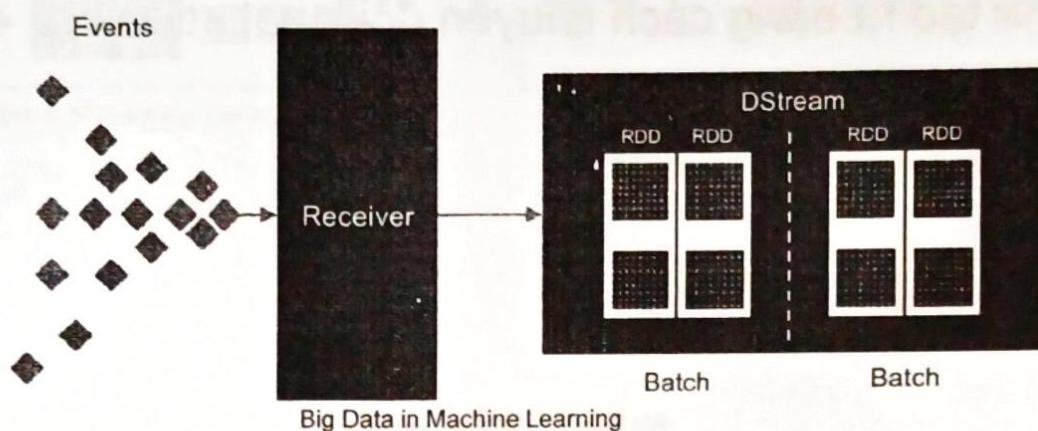
- Trong nội tại, một DStream được thể hiện bằng một chuỗi RDD liên tục và mỗi RDD chứa dữ liệu từ một khoảng nhất định.



Spark Streaming Fundamentals

• Input DStreams

- Là các DStream đại diện cho luồng dữ liệu đầu vào (input data stream) nhận được từ các streaming source.
- Mỗi input DStream liên kết với một Receiver object nhận dữ liệu từ một nguồn và lưu trữ nó trong bộ nhớ Spark để xử lý.



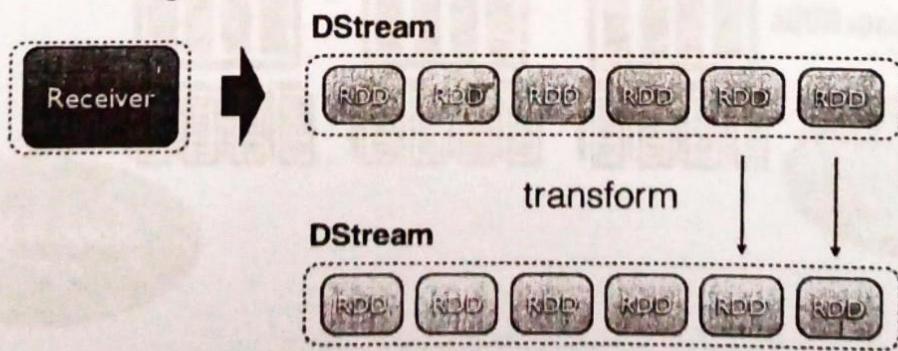
19

Spark Streaming Fundamentals



• Transformations

- Bất kỳ operation nào được áp dụng trên DStream đều chuyển sang các operation trên RDD bên dưới. Các phép biến đổi (transformation) cho phép dữ liệu từ input DStream đầu vào được hiệu chỉnh tương tự như RDD.
- DStreams hỗ trợ nhiều phép biến đổi có sẵn trên Spark RDDs thông thường.

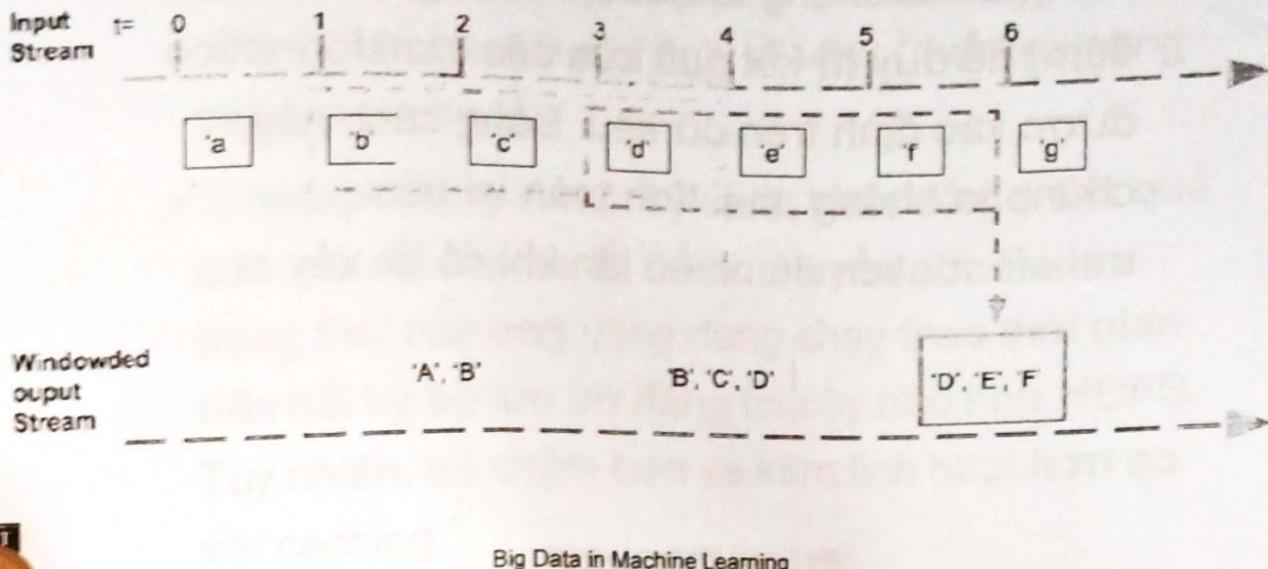


Big Data in Machine Learning

20

Spark Streaming Fundamentals

- Một số transformation trên DStream: map, flatMap, filter, reduce, groupBy
- Ví dụ:



Big Data in Machine Learning

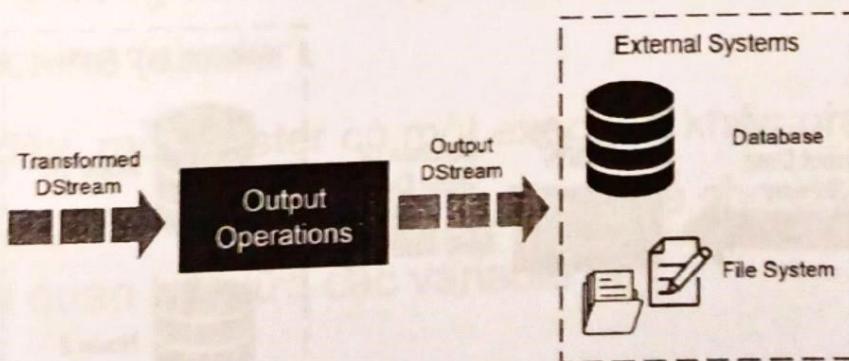
21

Spark Streaming Fundamentals



• Output Dstreams

- Các output operation cho phép dữ liệu DStream được đưa ra các external system như database hoặc file system. Các Output operation kích hoạt việc thực thi của tất cả các DStream transformation.



Spark Streaming Fundamentals

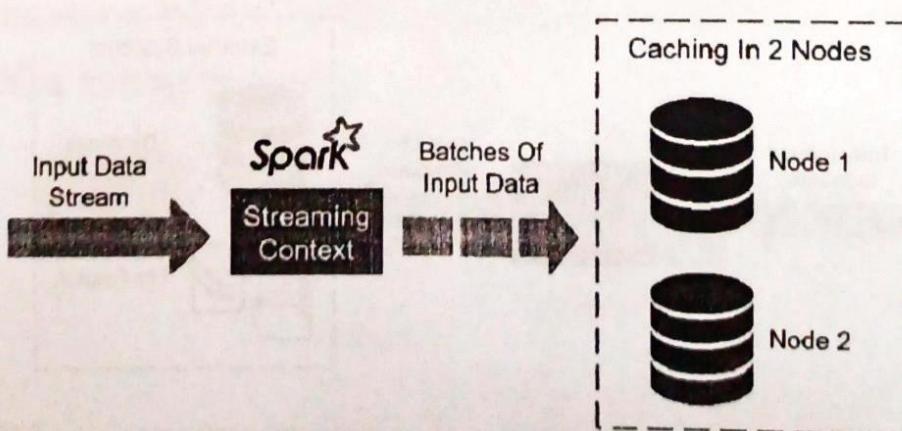
• Caching

- Catching một cách để đối phó với thử thách duy trì kết quả. Chúng ta có thể tạm thời lưu trữ các kết quả mà chúng ta đã tính toán (lưu vào bộ nhớ đệm) để duy trì kết quả của các transformation được xác định trên dữ liệu. Bằng cách này, chúng ta không phải tính toán lại các transformation đó nhiều lần khi có lỗi xảy ra.



Spark Streaming Fundamentals

- DStreams cho phép các developer lưu trữ dữ liệu cần truyền trong bộ nhớ. Điều này hữu ích nếu dữ liệu trong DStream sẽ được tính toán nhiều lần.
- Có thể được thực hiện bằng cách sử dụng phương thức persist() trên DStream.



Spark Streaming Fundamentals

• Checkpointing

- Caching cực kỳ hữu ích khi chúng ta sử dụng đúng cách nhưng nó đòi hỏi rất nhiều bộ nhớ. Và không phải ai cũng có hàng chục/ hàng trăm máy với RAM 128 GB để lưu trữ mọi thứ. Vì vậy, Checkpointing sẽ giúp chúng ta.
- “Checkpointing là một kỹ thuật khác để giữ kết quả của các dataframe đã được chuyển đổi. Nó lưu trạng thái của ứng dụng đang chạy theo thời gian trên bất kỳ bộ lưu trữ đáng tin cậy nào như HDFS. Tuy nhiên, nó chậm hơn và kém linh hoạt hơn so với caching.”

Big Data in Machine Learning

25

Spark Streaming Fundamentals

□ Shared Variable

- Đôi khi chúng ta cần định nghĩa các function như map, reduce, filter cho ứng dụng Spark thì phải được thực thi trên nhiều cluster. Các variable được sử dụng trong function này được sao chép vào từng machine (cluster).
- Ở đây, mỗi cluster có một executor khác nhau và chúng ta muốn một thứ gì đó có thể cho chúng ta một mối quan hệ giữa các variable này.

Spark Streaming Fundamentals

- Ví dụ: Giả sử ứng dụng Spark đang chạy trên 100 cluster khác nhau lấy ảnh Instagram được đăng bởi những người từ các quốc gia khác nhau. Chúng ta cần đếm một số lượng thẻ (tag) cụ thể đã được đề cập trong một bài đăng (post).
- Nay, mỗi executor của từng cluster sẽ tính toán kết quả của dữ liệu hiện tại trên cluster cụ thể đó. Nhưng chúng ta cần một thứ gì đó giúp các cluster này giao tiếp với nhau để có thể nhận được kết quả tổng hợp. Trong Spark, chúng ta có các shared variable cho phép thực hiện công việc này.



Spark Streaming Fundamentals

☐ Accumulator Variable

- Là các biến chỉ được thêm vào thông qua associative & commutative operation, để hiện thực việc liên kết và giao hoán. Ví dụ: sum & maximum sẽ làm việc, trong khi mean thì không.
- Theo dõi các accumulator trong UI có thể hữu ích để hiểu tiến trình của các running stage. Spark hỗ trợ numeric accumulator. Chúng ta có thể tạo các accumulator có đặt tên hoặc không.



Spark Streaming Fundamentals

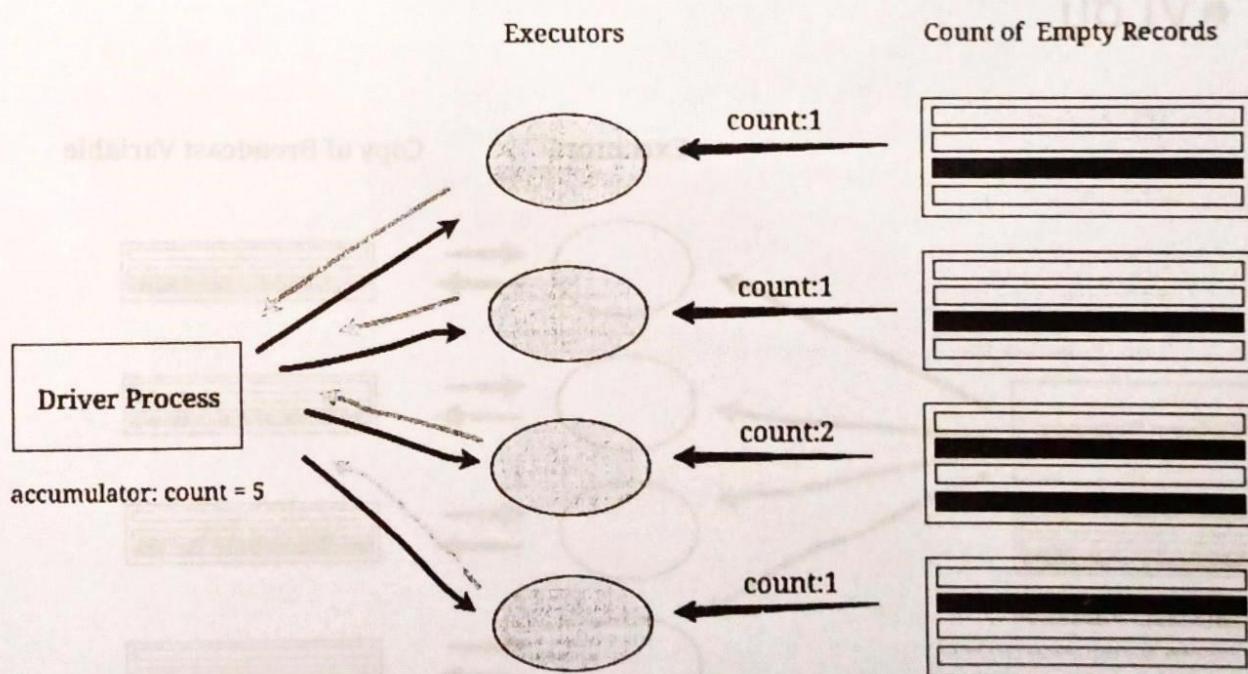


- Ví dụ: Một số usecase như số lần xảy ra lỗi, số lượng blank log, số lần nhận request từ một quốc gia cụ thể... Tất cả đều có thể giải quyết bằng cách sử dụng accumulator. Executor trên mỗi cluster gửi dữ liệu trở lại driver process để cập nhật các giá trị của các accumulator variable.

Big Data in Machine Learning

29

Spark Streaming Fundamentals



Spark Streaming Fundamentals

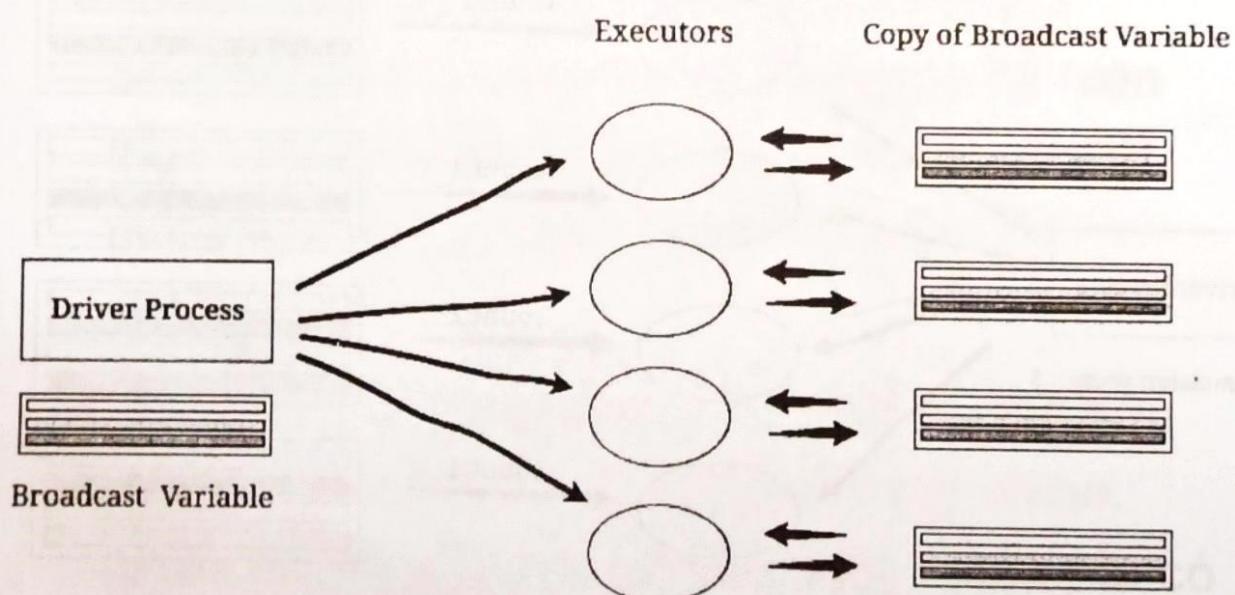
❑ Broadcast Variable

- Broadcast variable cho phép lập trình viên giữ một read-only variable được lưu trong bộ nhớ cache trên mỗi máy thay vì gửi một bản sao của nó với các task. Chúng có thể được sử dụng để cung cấp cho mỗi node một bản sao của một tập dữ liệu đầu vào lớn một cách hiệu quả. Spark cũng cố gắng phân phối các broadcast variable bằng broadcast algorithm hiệu quả để giảm chi phí liên lạc (communication cost).



Spark Streaming Fundamentals

• Ví dụ



Nội dung

-
1. Giới thiệu Spark Streaming
 2. Spark Streaming Fundamentals
 3. Làm việc với PySpark Streaming



Làm việc với PySpark Streaming

<http://localhost:8080/notebooks/Chapter10/Spark%20Streaming%20with%20Python.ipynb>





Chapter 10: Introduction to Spark Streaming

Note on Streaming

Streaming is something that is rapidly advancing and changing fast, there are many new libraries every year, new and different services always popping up, and what is in this notebook may or may not apply to you. Maybe you're looking for something specific on Kafka, or maybe you are looking for streaming about Twitter, in which case Spark might be overkill for what you really want. Realistically speaking each situation is going to require a customized solution and this course is never going to be able to supply a one size fits all solution. Because of this, I wanted to point out some great resources for Python and Spark Streaming.

- The Official Documentation is great. This should be your first go to. (<http://spark.apache.org/docs/latest/streaming-programming-guide.html#spark-streaming-programming-guide>)
- Fantastic Guide to Spark Streaming with Kafka (<https://www.rittmanmead.com/blog/2017/01/getting-started-with-spark-streaming-with-python-and-kafka/>)
- Another Spark Streaming Example with Geo Plotting (<http://nbviewer.jupyter.org/github/ibm-cds-labs/spark.samples/blob/master/notebook/DashDB%20Twitter%20Car%202015%20Python%20>)

Let's discuss SparkStreaming!

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like Kafka, Flume, Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like map, reduce, join and window. Finally, processed data can be pushed out to filesystems, databases, and live dashboards. In fact, you can apply Spark's machine learning and graph processing algorithms on data streams.





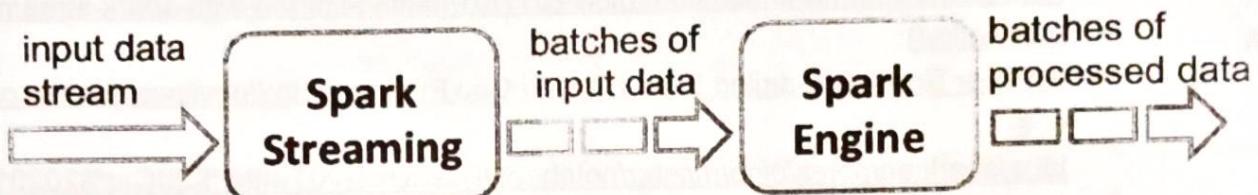
Keep in mind that a few of these Streaming Capabilities are limited when it comes to Python, we need to reference the documentation for the most up to date information. Also the streaming contexts tend to follow more along with the older RDD syntax, so a few things might seem different than what we are used to seeing, keep that in mind, you'll definitely want to have a good understanding of lambda expressions before continuing with this!

There are SparkSQL modules for streaming:

<http://spark.apache.org/docs/latest/api/python/pyspark.sql.html?highlight=streaming#module-pyspark.sql.streaming> (<http://spark.apache.org/docs/latest/api/python/pyspark.sql.html?highlight=streaming#module-pyspark.sql.streaming>)

But they are all still listed as experimental, so instead of showing you something that might break in the future, we'll stick to the RDD methods (which is what the documentation also currently shows for streaming).

Internally, it works as follows. Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches.



Twitter Example

In order to use all of this though, we need to setup a Developer API account with Twitter and create an application to get credentials. Review the video for instructions on how to do this or if you are already familiar with it, just get the credentials from:

<https://apps.twitter.com/>

Once you have that you also need to install python-twitter, a python library to connect your Python to the twitter dev account.

You probably won't be able to run this example and then previous in the same notebook, you need to restart your kernel.

Let's get started!

Begin by running the TweetRead.py file. Make sure to add your own IP Address and your credential keys.

In [40]: `import findspark
findspark.init()`



In [41]: `import pyspark`

In [42]: `# May cause deprecation warnings, safe to ignore, they aren't errors`
`from pyspark import SparkContext`
`from pyspark.streaming import StreamingContext`
`from pyspark.sql import SQLContext`
`from pyspark.sql.functions import desc`

In [43]: `# Can only run this once. restart your kernel for any errors.`
`sc = SparkContext()`

In [44]: `ssc = StreamingContext(sc, 10)`
`sqlContext = SQLContext(sc)`

In [45]: `socket_stream = ssc.socketTextStream("127.0.0.1", 5555)`

In [46]: `lines = socket_stream.window(20)`

In [47]: `from collections import namedtuple`
`fields = ("tag", "count")`
`Tweet = namedtuple('Tweet', fields)`

In [48]: `# Use Parenthesis for multiple lines or use \.`
`(lines.flatMap(lambda text: text.split(" "))) # Splits to a List`
`.filter(lambda word: word.lower().startswith("#")) # Checks for hashtag call`
`.map(lambda word: (word.lower(), 1)) # Lower cases the word`
`.reduceByKey(lambda a, b: a + b) # Reduces`
`.map(lambda rec: Tweet(rec[0], rec[1])) # Stores in a Tweet Object`
`.foreachRDD(lambda rdd: rdd.toDF().sort(desc("count")) # Sorts Them in a DF`
`.limit(10).registerTempTable("tweets"))) # Registers to a table.`

Now run TweetRead_new.py

- ..\\Spark_Streaming>python TweetRead_new.py >> tweets_vCov.txt

In [50]: `ssc.start()`

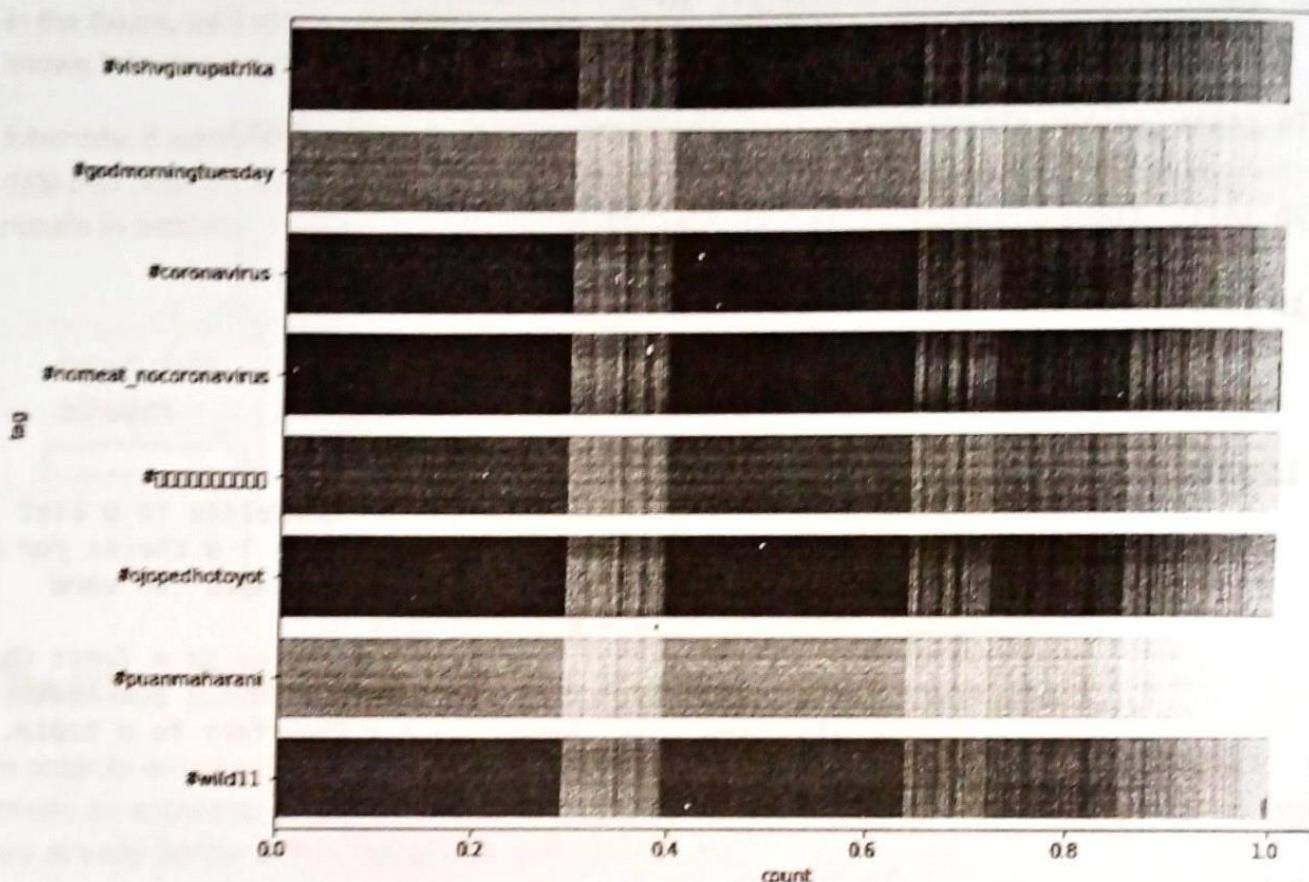
In [51]: `import time`
`from IPython import display`
`import matplotlib.pyplot as plt`
`import seaborn as sns`
`# Only works for Jupyter Notebooks!`
`%matplotlib inline`

In [55]: try:

```

        count = 0
        while count < 10:
            time.sleep( 3 )
            top_10_tweets = sqlContext.sql( 'Select tag, count from tweets' )
            top_10_df = top_10_tweets.toPandas()
            display.clear_output(wait=True)
            plt.figure( figsize = ( 10, 8 ) )
            sns.barplot( x="count", y="tag", data=top_10_df)
            plt.show()
            count = count + 1
    except:
        print("No tweets now")

```



In [56]: ssc.stop()

Chapter 10: Spark Streaming

Ex1: Pre-processing Data from Tweets

Requirement:

- Read data from file (Tweets)
- Pre-process data
- Save data after pre-processing to new file.

```
In [1]: # pip install textblob  
import csv  
from textblob import TextBlob
```

```
In [2]: import pandas as pd
```

```
In [3]: tweetdata = 'tweets_christmas.txt'  
sentences = []  
sentiment_polarity = []  
sentiment_subjectivity = []
```



```
In [4]: with open(tweetdata, 'r') as csvfile:  
    rows = csv.reader(csvfile)  
    for row in rows:  
        sentence = row[0]  
        blob = TextBlob(sentence)  
        if ("Error on_data" not in sentence):  
            print (sentence)  
            print (blob.sentiment.polarity, blob.sentiment.subjectivity)  
            sentences.append(sentence)  
            sentiment_polarity.append(blob.sentiment.polarity)  
            sentiment_subjectivity.append(blob.sentiment.subjectivity)  
  
b'Christmas is here \xf0\x9f\x8e\x84\xf0\x9f\x8e\x81\xf0\x9f\x8e\x84 Beautiful  
1 Christmas tree \xf0\x9f\x8e\x84'  
0.85 1.0  
b'RT @VrglSullano: All I want for Christmas is cashhhhhhhhhhhhhhhhhhhhhhhhhhhhh  
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh  
0.0 0.0  
b'RT @_mymusictaste: \xf0\x9f\x93\xab ATINY! Surprise! \xf0\x9f\x8e\x86\nChris  
tmas is starting earlier this year! \xf0\x9f\x8e\x81\n\nWelcome ATEEZ GLOBAL  
FANSIGN EVENT IN LOS ANGELES. \xf0\x9f\x8e\xb6\nSta\xe2\x80\xab'  
0.0 0.19999999999999998  
b'RT @GrowLevel: \xf0\x9f\x8e\x84Merry Christmas\xf0\x9f\x8e\x84\n\xe3\x83\xab  
\xe3\x83\xad\xe3\x83\xb3\xe3\x82\xb1\xe3\x83\xbc\xe3\x82\xad\xe3\x82\x92\xe6  
\x8a\xbd\xe9\x81\xb8\xe3\x81\xa7\xe5\x90\x8d\xe6\xa7\x98\xe3\x81\xab\xe3\x83  
\x97\xe3\x83\xac\xe3\x82\xbc\xe3\x83\xb3\xe3\x83\x88\xf0\x9f\x8e\x81\xf0\x9f  
\x8d\x88\n\xe5\x95\x86\xe5\x93\x81\xe3\x81\xaf\xe3\x82\xaf\xe3\x83\xbc\xe3  
\x83\xab\xe4\xbe\xbf\xe3\x81\xa7\xe3\x81\x8a\xe5\xb1\x8a\xe3\x81\x91\xe3\x81  
\x97\xe3\x81\xbe\xe3\x81\x99\xe3\x80\x82\n\xe2\x80\xbb\xe5\xab\xe4\x98\xab  
\xe3\x81\x8b\xe3\x82\x82\x892\xe6\x97\xab\xe4\xbb\xab\xe4\xb8\x8a\xe3\x81\x8b\xab  
\x81\x8b\xe3\x82\x8b\xe5\x9c\xb0\xe5\x9f\x9f\xe3\x81\xaf\xe4\xb8\x8d\xe5\x8f  
\xaf\xe3\x80\x82\n\xe5\xhf\x9c\xe5\x8b\x9f\xe7\xb7\xab\xe5\x88\x87\xe3\x82\x82
```

```
In [5]: data = pd.DataFrame({"sentence": sentences,
                           "sentiment_polarity":sentiment_polarity,
                           "sentiment_subjectivity":sentiment_subjectivity
                           })
```

```
In [6]: data = data.drop([0, 1])
```

```
In [7]: data['sentence'] = data['sentence'].str.replace("b'", "")
```

```
In [8]: data.head()
```

Out[8]:

	sentence	sentiment_polarity	sentiment_subjectivity
2	Decorating Sebastians Grave Christmas 2019 VI...	0.0	0.0
3	RT @_mymusictaste: \xf0\x9f\x93\x a3ATINY! Surp...	0.0	0.2
4	Christmas partyyyy'	0.0	0.0
5	Working on a maaaybe Christmas-y themed piece....	0.5	0.6
6	Coastes shenanigans pre-Christmas celebrations...	0.0	0.0

In [9]: `data.to_csv("tweets_christmas.csv")`

Another solution: Build function to read txt file and convert to csv file

```
In [10]: def read_and_pre_pro(file_in, file_out):
    sentences = []
    sentiment_polarity = []
    sentiment_subjectivity = []
    with open(file_in, 'r') as csvfile:
        rows = csv.reader(csvfile)
        for row in rows:
            sentence = row[0]
            blob = TextBlob(sentence)
            if ("Error on_data" not in sentence):
                #print (sentence)
                #print (blob.sentiment.polarity, blob.sentiment.subjectivity)
                sentences.append(sentence)
                sentiment_polarity.append(blob.sentiment.polarity)
                sentiment_subjectivity.append(blob.sentiment.subjectivity)
    data = pd.DataFrame({"sentence": sentences,
                         "sentiment_polarity":sentiment_polarity,
                         "sentiment_subjectivity":sentiment_subjectivity
                        })
    data.sentence = data.sentence.str.replace("b'", "")
    data.to_csv(file_out)
```

```
In [11]: file_in = "tweets_football.txt"
file_out = "tweets_football.csv"
read_and_pre_pro(file_in, file_out)
```

```
In [12]: df = pd.read_csv("tweets_football.csv", index_col=0)
```

```
In [13]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1333 entries, 0 to 1332
Data columns (total 3 columns):
sentence           1333 non-null object
sentiment_polarity 1333 non-null float64
sentiment_subjectivity 1333 non-null float64
dtypes: float64(2), object(1)
memory usage: 41.7+ KB
```



In [14]: df.head()

Out[14]:

	sentence	sentiment_polarity	sentiment_subjectivity
0	Listening on port: 5555	0.00	0.0
1	Received request from: ('127.0.0.1'	-0.75	1.0
2	Listening on port: 5555	0.00	0.0
3	Received request from: ('127.0.0.1'	-0.75	1.0
4	EVA SOCCER memenuhi keperluan football dan fut...	0.00	0.0

In [15]: indexNames = df[df['sentence'].str.contains("Listening on port")].index
Delete these row indexes from DataFrame
df = df.drop(indexNames)

In [16]: df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1331 entries, 1 to 1332
Data columns (total 3 columns):
sentence           1331 non-null object
sentiment_polarity 1331 non-null float64
sentiment_subjectivity 1331 non-null float64
dtypes: float64(2), object(1)
memory usage: 41.6+ KB
```

In [17]: df.head()

Out[17]:

	sentence	sentiment_polarity	sentiment_subjectivity
1	Received request from: ('127.0.0.1'	-0.75	1.000000
3	Received request from: ('127.0.0.1'	-0.75	1.000000
4	EVA SOCCER memenuhi keperluan football dan fut...	0.00	0.000000
5	RT @AndrewMLind: Per ESPN	0.00	0.000000
6	RT @ANNMediaSports: USC football Head coach Cl...	0.00	0.333333