Twitter #Hashtag Recommendation System -- Project Report

MSBA 6330 – Gold Team #1:

Bhuvan Oberoi
Meg Ou
Peiwen Zhao
Sumit Verma
Utkarsh Khandelwal

Introduction:

We live in a connected world and information around us is not isolated but rich, connected. Only a database that embraces these relationships is able to store, process and query the data efficiently. Databases like relational databases which provide us with ACID guarantees, compute these relationships expensively, even with proper indexing when data gets huge, operations like JOIN operation required to query the data, take time to run and often lead to spool space issues and thus fail to perform efficiently.

As data grows the innovations to store, analyze and process the data also come into play. ACID properties of the relational database though provided us consistency and guarantees on our data and often slowed down our databases by the inherent requirements like normalization that come along with it. Hence, they were the first one to go out. Thus came noSQL revolution. The data was stored in distributed systems and was stored as key-value pairs. As we grew more advanced we started storing this data into documents in the form of dictionaries which are essentially key values, but indexing could be performed on them.

But one key thing this was missing was that this kind of data was missing relationships between our data. This is where graph databases come in. with graph databases we can store these relations and access these relations easily in an efficient manner. This is important as the relations in the data have meanings. For example, fraud detection systems there are relationships between these data items that can indicate interesting patterns in the data potentially useful for mining these interesting relationships. Independent of the total size of the dataset, graph databases excel at managing highly connected data and complex queries.

Sponsored by Neo, Neo4j is an open-source NoSQL graph database implemented in Java and Scala. It is used today by many companies and organizations. Use cases include

Github repository:

https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System

matchmaking, network management, software analytics, scientific research, routing, organizational and project management, recommendations, social networks, and more.

In this project we leveraged the power of graph databases and built a twitter hashtag recommendation system that uses the cypher querying language that runs on Spark GraphX to provide real-time #hashtag recommendations for twitter by traversing over a large number of components. We combined graph technology and the power of Spark installed on an AWS cluster that ensuring scalability of the system.

Process Implementation:

Components:

The process for creating the recommendation system is divided into the following components:

Data aggregation:

We start the process by downloading tweets randomly through a twitter streaming API. We downloaded the 1.4 million tweets which we filtered for english language. This led to a collection of 289000 tweets. The data collection aggregated two days' worth of tweets captured through the twitter API (random selection of 1% of the total tweets on twitter).

Data Pre-processing:

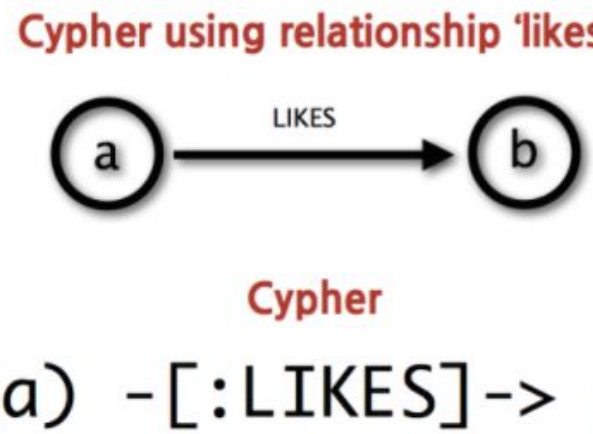
We then pre-process the data in R on local system and upload the data onto an AWS EC2 cluster for Neo4j. This graph can be accessed through the neo4j browser. On the AWS cluster we installed spark and jupyter notebook with Scala to run and process the data using Spark GraphX.

Neo4J Graph Database:

Neo4j Graphs can be accessed through a Graph Query Language called Cypher developed by neo4j itself. Cypher is a declarative, SQL-inspired language for describing patterns in graphs visually using an ascii-art syntax. A simple example of a Cypher query can be seen from the following diagram:

Github repository:

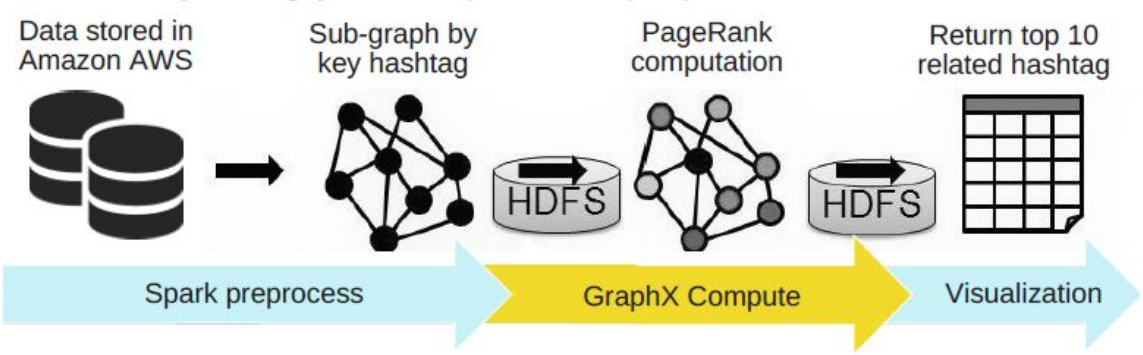
https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System



Here a and b enclosed in parentheses are called nodes and the relationship is represented in square [] brackets.

Apache Spark GraphX:

For running our queries we used Spark GraphX environment that we set up on an AWS EC2 cluster. Following is the process diagram for the interactions among different technologies in the environment.

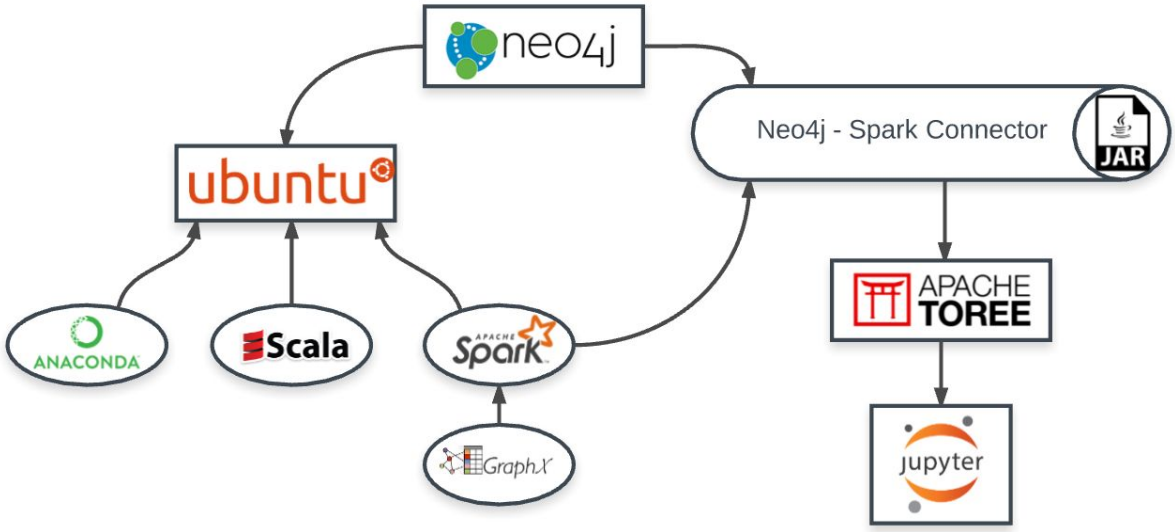


Github repository:

https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System

Part A - Special Topics on Big Data

Conceptual Foundation:



AWS:

The secure cloud services platform, it provides us an environment to store our large database and apply analysis on it.

Neo4j:

Neo4j is a graph database that takes connection-first approach data with persisting relationships through every transition of existence, in our case, are the tweets, hashtags, and user ID.

Spark GraphX:

Spark GraphX is an Apache Spark's API for graph and graph-parallel computation. It retains Spark's flexibility, fault tolerance, and ease of use with fast graph processing system at the same time.

Spark Scala:

Scala is an acronym for "Scalable Language" which is developed as an object-oriented programming language. It's style is similar to a scripting language.

Apache Toree:

A notebook provider which allows applications connect with Spark Scala kernel interactively without the limitations like bundling with the JAR.

Github repository:

https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System

Demo/Example/Proof of Concept:

Here is a sample demo for our project, step by step tutorial please check our github repository:
https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System

Neo4j database demo:

As shown in the demo below, each circle represent a node in the twitter structure.

- Red circle: Twitter user
- Green circle: Tweet (contains tweet id and tweet context)
- Blue circle: Hashtag

Multiple tweets are linked together as they may contain common hashtag and the most trendy and popular hashtag (which being mentioned in many tweets becomes the central node of a tweet's community)

Spark Scala demo:

After implementing neo4j & Spark connector on AWS, now we are able to write query in Spark Scala to utilize the scalability to speed up Hashtag recommendation system.

Github repository:

https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System

Sample code:

```
In [1]: import org.neo4j.spark._
        val neo = Neo4j(sc)

In [5]: val rdd = neo.cypher("""MATCH (h:Hashtag)<-[:HASHTAG]-()-[:HASHTAG]->(h2:Hashtag)
        WHERE h.hashtag = '#apple' RETURN h2.hashtag, COUNT(*) AS cnt ORDER BY cnt DESC""").loadRowRdd
        val x = rdd.collect()

In [6]: import Array._

        print ("  Count  |    Hashtag  \n|---------------------------------| \n")
        for (i <- 0 to 10) {

               print ("   ")
               print(x(i)(1))
               print ("    |       ")
               print(x(i)(0))
               print ("    ")
               println
               print ("|---------------------------------|")
               println
             }
```

```
  Count  |    Hashtag
|---------------------------------|
   2    |       #samsung
|---------------------------------|
   2    |       #samsungnote
|---------------------------------|
   2    |       #ios11
|---------------------------------|
   2    |       #battery
|---------------------------------|
   2    |       #ios
|---------------------------------|
   2    |       #iphonex
|---------------------------------|
```

Part B - Analytics with Big Data

Background:

Nowadays, most of the social networks are offering the function of hashtag "auto-completed" system. For example, on Instagram if you type "fo", the recommendations will be like "food", "foodie", "football" etc,; however, there are no recommendations for the next hashtag so far. For the user to become part of an intended hashtag social group, it requires the user to write exactly the right case and spelling in the hashtags. That is the reason why we are looking to create a real-time hashtag recommendation system that finds the most relevant and trending hashtags based on user's first hashtag as input. With the recommended following hashtag, a user can increase his public exposure more easily by joining his post into the right hashtag's streaming Twitter or Instagram.

Our goal is to find the hashtag list that has not only strong relations with the user's first input hashtag but also the most popular to offer the most popular hashtags by analyzing pre-streamed Tweets data on AWS. The Neo4j database can graphically and intuitively help us to find the connected nodes- the hashtags- in the sub-group.

Github repository:

https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System

Data:

We use Python to scrape high volume Twitter data into Neo4j database on AWS EC2. The Twitter text data was transformed into “nodes” and “relations” format to input to Neo4j, we can then run Spark and Scala to query the data and do the computing on Spark installed in the EC2. Our raw data contained 1.4 million Tweets, from which we filtered for English text Tweets, which included the content of the Tweet, the Hashtags they mentioned, and the user ID. After cleaning the data, we ended up with 0.3 million tweets.

Methodology:

We started by pushing data through R into Neo4j database which is launched on AWS EC2 because we need cloud computing to deal with large amount of data. Then we analyzed and visualized the data through Spark and Scala. First, place user input as the first hashtag node into our query, then we can find the relations through the “lines” between nodes. Each line indicates relationship like “re-tweet” and “Hashtag”. Finally we search into sub-graph which is connected by this line then do the calculation.

Cypher query language code for hashtag recommendation:

```
MATCH p=(h:Hashtag {hashtag:'#ios'})-[r:HASHTAG]-()-[:HASHTAG]-(h2)
return count(r) as cnt, h2 order by cnt desc
```

Findings:

Each hashtag will have different sub-graph output in the Neo4j browser based on the all the Tweets that have the specified Hashtag. In the sub-graph, the Tweet node shows the relationship of hashtags with one another. We can choose these hashtags then sort them by the degree of popularity by doing a simple order by query on Spark GraphX.

As far as the recommendations for Hashtags on Twitter go, we got some really interesting results on the recommendations that we could gather from a simple Cypher query.

Post, performing the above methodology, we found out that the queries were amazingly fast due to the nature of graph databases to capture the relationships among the nodes. Graph databases maintain an index for all the nodes and relationships are just pairs of indexes. So when a query is launched, the database gets subsetted on this index and a relatively much smaller data is computed.

We also compared the speed of these queries on our local system vs AWS cluster, there was a big difference in the query execution times.

Last but not the least, we processed the queries in spark GraphX, and ran the same queries through spark to ensure scalability of the above process. The queries, as expected were lightning fast and led us to believe that graph databases are actually efficient for databases with inherent relations. Also, as we have already established as a POC, this

Github repository:

https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System

technology is actually scalable for huge databases as it can run on parallel distributed systems.

Concluding Remarks

Our project is analyzing Twitter streaming data in Neo4j graphical database on AWS EC2 instance by Spark GraphX; which allow us to build Hashtag recommendation system that suggests the user best following Hashtag.

The long term expectation for this system is to implement it while user is composing the new post. This could be broadly utilize in social media like Twitter and Instagram. User with the suggested Hashtags can join the desired grouped-posts more easily; it leads to the increase of the customer experience and exposure. Further business implementation includes building a highly scalable product based recommendation system and understanding what people are associating a particular Hashtag, which could be a company or a product, with together a high-level sentiment around it.

Through this project, we had learned the findings by hands-on experiences. We compared the effectiveness by running all the possible solutions. Although we faced many difficulties during this project since is a new technology we never had experience before, we thoroughly enjoyed the process of exploration.

References and the resources:

https://neo4j.com/product/

https://spark.apache.org/graphx/

https://www.analyticsvidhya.com/blog/2017/01/scala/

https://toree.apache.org

https://medium.com/@josemarcialportilla/getting-spark-python-and-jupyter-notebook-running-on-amazon-ec2-dec599e1c297

https://toree.apache.org/docs/current/user/installation/

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html?icmpid=docs_ec2_console

Github repository:

https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System

https://stackoverflow.com/questions/40831991/jupyter-apache-toree-scala-kernel-is-busy

Appendix:

Github repository:

https://github.umn.edu/zhao0885/Bigdata-project-Hashtag-Recommendation-System