Take Home Exam of Data Engineering

Yuhao Cheng, 1379879

July 1, 2019

1 Lecture 1

Question: Compare and contrast Big data enthusiasm and Big data skepticism. Include a discussion of current debates, including ONeils essay and more recent developments in (ir)responsible and (un)fair data science.

Answer: With the increasing volume and detail of information captured by enterprises and people, the rise of multimedia, social media, and the Internet of Things have fuel exponential growth in data. The increasing accumulation of big data allow humans to discover patterns, the more data you can get, the more value you can mine, and it seems that predicting future is no longer only a mind-reading in science fiction movies. Many people believe that big data will become a key basis of competition, underpinning new waves of productivity growth, innovation, and consumer surplusas long as the right policies and enablers are in place. McKinsey studied big data in five domainshealthcare, public sector, retail, manufacturing and personal-location data, they conclude that Big data can generate great value in each of these domains.

However, some people call for skepticism for big data. They said most business need only moderately complex data in moderate quantity. Also, many companies dont need Big Data. Google needs Big Data to index the World Wide Web, to map the Earth, and to make self-driving cars function. But most organizations dont have such extreme needs. Moreover, Big data could be misleading and dangerous, people could only understand what the presenter is trying to convey instead of looking deeper into what the data is actually showing, this might even lead to mistakes in decision-making of organizations. Regarding the security of data, it has been discussed continuously in recent years. Cambridge analysts use Facebook's data to influence the US presidential election as an example. As for the seIn ONeils essay, the author said we need to find a place inside business for skepticism. Although she was calling for finding a place for skepticism, she have no easy formula for how to achieve it.

Considering the debate of Big Data, the right way is to choose a middle-of-the-road approach to deal with big data: dont be blindly optimistic, dont be blindly pessimistic. In some parts of several organizations, those strategies should address Big Data, either as a helpful tool to reach goals, or as a challenge that must be addressed along the way. For these organizations, The biggest challenge is finding the right sort of people that can perform this kind of work and find the answers they're looking for. However, Big Data should not be a goal, and for many organizations, it is neither a necessary tool nor a pressing challenge. It is not something that every business needs. For many organizations theres nothing vital about Big Data. Instead, they should focusing on their own business goals and obligations.

People and organization should focus on fairness and transparency, in order to develope responsible data science. Fairness and transparency are important themes in data science because they are important themes to society. They are basic to human rights as well as to ethics in society. Transparency implies knowledge about why an automatic system took a decision. Then the person affected by that decision can analyze the reasons behind it and decide to participate it or not. To define what is fair, we can have a functional definition by saying that if the consequences are unintended and hurt people, then the data science used was probably unfair.

In ONeils essay, the author told us that data is a tool, and like any other tool, its neither good nor evil; that depends on how its used. youre more likely to use data effectively, and understand how other people are using it, if you develop a healthy skepticism, which is to say a healthy habit of mind to questionand insist on understandingthe arguments behind the conclusions.

2 Lecture 2

Question: Is the databases community critique on MapReduce addressed by recent progress, e.g., Spark, Flink, etc.? What remains to be done? Include a discussion on Stonebreaker/De-Witts paper and more recent work.
Answer: In the paper MapReduce: A major step backwards, the author gave a lot of concerns and criticism about MapReduce. 1. MapReduce is a step backwards in database access, it learned nothing of schema and High-level access languages from lessons in the 40 years since IBM first released IMS in 1968. 2. MapReduce is a poor implementation. it has no indexes and therefore has only brute force as a processing option. 3. MapReduce is not novel, it is more than 20 years old. 4. MapReduce is missing features. It misses features like bulk loader, indexing, updates, transactions, etc. 5. MapReduce is incompatible with the DBMS tools. It cannot use tools like Report writers, Business intelligence tools, Data mining tools, etc. and has none of its own. Thus, the author think much larger community engaged in the design and implementation of scalable query processing techniques is exciting. In fact, in addition to the authors criticism, with the development of new progress in recent years, such as spark and Flink, it is easy to see the criticism and complains of MapReduce everywhere on the Internet.

Apache Spark is a framework for executing general data analytics over distributed system and computing clusters, for example Hadoop. Apache Spark does in-memory computations with higher speed, low latency data process on MapReduce. Here are some differences between the Spark and MapReduce: 1, Spark speed is faster than MapReduce, Spark stores the intermediate data of the operation in memory, and iterative calculation efficiency is higher; the intermediate result of mapreduce needs to be saved to disk, which affects performance; 2, spark fault tolerance, it achieves efficient fault tolerance through flexible distributed data set RDD;3, spark is more general, spark provides multiple functional APIs of transformation and action, in addition to streaming processing sparkstreaming module, graph computing GraphX, etc.; mapreduce only provides map and reduce two operations, computing framework (API) has limitations; 4, spark framework and ecology is more complicated; mapreduce framework and its ecology is relatively simple, the performance requirements are relatively weak, but the operation is relatively stable, suitable for long-term background operation; 5. Scalable Language is the best language for parallel computing. Compared with Java, it greatly reduces the amount of code (many parts of the Hadoop framework are written in Java).

Apache Spark could replace Hadoop MapReduce but Spark needs a lot more memory; however MapReduce kills the processes after job completion; therefore it can easily run with some in-disk memory. Apache Spark performs better with iterative computations when cached data is used repetitively. As a conclusion Hadoop MapReduce performs better with data that doesnt fit in the memory and when there are other services to be executed. While Spark is designed for instances where data fits in the memory especially on dedicated clusters. In terms of ease of use Hadoop MapReduce is written in Java and is difficult to program, whereas Apache Spark has flexible and easy to use APIs in languages like Python, Scala and Java. Developers can write user-defined functions in Spark and even include interactive mode for running commands.

There are also some problems with these new progress, for instance, it is expensive, in-memory capability can become a bottleneck when we want cost-efficient processing of big data as keeping data in memory is quite expensive; no file management System; higher latency, costly when transferring data between systems, since they have their own development tool and languages.

Considering the flexibility, speed and ease of using new tools like spark, they are expected to be adopted more widely and largely replace MapReduce. But there would be still some areas where MapReduce would be required, particularly when non-iterative computations is required with limited memory availability.

3 Lecture 4

Question: Compare and contrast at least three different semantics for subgraph pattern matching queries which have been proposed (e.g., homomorphism, isomorphism, similarity/simulation, ...).
Answer: Graphs are widely used to model complicated data semantics in many applications in bioinformatics, chemistry, social networks, pattern recognition, etc. Graph pattern match-

ing query is to identify all occurrences of a small query graph in a large target graph. We often query graphs for certain patterns, for example retrieving all patients and their friends. These queries are so called containment queries and find all subgraphs that match certain patterns in a graph. With the prevalence of large-scale networks (having millions of nodes and billions of edges), such as knowledge graphs, social networks, XML and web graphs, protein interaction and brain networks, the problem of effectively and efficiently finding all occurrences of a small query graph in a large target network is of high importance. Graph pattern matching can be defined via a strict notion of subgraph isomorphism, which is an injective function from the nodes in the query graph to the nodes in the target graph, and allows one-to-one matching between the nodes and edges. While the decision version of subgraph isomorphism is NP-complete; in reality, the problem could be solved in polynomial time (in size of target graph).

Here we introduce three different subgraph pattern matching semantics are isomorphism , cyphermorphism and homomorphism. Basically, under the isomorphism semantic, two different query vertices or edges may not map to the same data vertices or edges. In the property graph model a distinction is sometimes made between node and edge isomorphism where the restriction only applies to vertices or edges respectively but not both. Semantic for multigraphs is undefined; under the Cyphermorphism semantic, two query edges may not map to the same data edge. Homomorphism does not need to comply with such restriction and may therefor result in larger graphs. From isomorphism , cyphermorphism and homomorphism, they are being less restrictive (more powerful); from homomorphism, cyphermorphism and isomorphism, they are going to be easier to get started with.

Presence of noise and inconsistencies in data lead to the application of so called similaririty queries that find all subgraphs that approximately match or are contained by a query. Among the various graph similarity measures used in these studies, graph edit distance has been widely accepted for representing distances between graphs. Compared with alternative distance or similarity measures, graph edit distance has three advantages: (1) It allows changes in both vertices and edges; (2) it reflects the topological information of graphs; and (3) it is a metric that can be applied to any type of graphs.

Here we discuss the Pros and cons of the different semantics:

1. They all have the same worst-case time complexity: O(n k) (n = num. data vertices, k = num. query vertices) However, if we apply iso/cypher-morphism to recursive path queries, things blow up Isomorphism and cyphermorphism have limitations

2. Homomorphism may return more matches than expected and requires additional non-equality constraints

3. Isomorphism and cyphermorphism dont translate as well to/from SQL

In conclusion, we introduced the Graph pattern matching queries, and discussed different subgraph pattern matching semantics, including isomorphism , cyphermorphism homomorphism and similaririty queries. We also compared the procs and cons of different semantics.

4 Lecture 5

Question: What is the state of the art in schema languages for JSON? Give a brief summary, with appropriate references to the scientific and industrial literature.
Answer: JSON is the abbreviation of JavaScript Object Notation. It is a simplified data exchange format. It is the most common exchange format for data exchange between Internet services, and it is simple and readable. Since it is a format for data exchange, there is data exchange between the two parties. How to agree or verify the data format of the other party meets the requirements, which becomes a problem that needs to be solved for service interaction. JSON Schema is a powerful tool for validating the structure of JSON data, it is a standard for defining json data constraints [1]. According to this convention mode, both parties exchanging data can understand the requirements and constraints of json data, and can also verify the data according to this to ensure the correctness of data exchange. JSON Schema is the general attempt to define a schema language for JSON documents, and it is slowly being established as the default schema specification for JSON. There have been other alternatives for defining schemas for JSON documents, but these are either based on JSON Schema itself or have been designed with a particular set of use cases in mind [2]. The latest version of Json Schema is draft 07, released on 2018-03-19, and the draft 08 version will be released in 2019.

JSON Schema itself is written in JSON. It is a specification of JSON data, not a computer program. It is just a declaration format that describes "the JSON data structure." This is both its strength and its weakness (similar to other Schema languages). It is easy to succinctly describe the surface structure of the data and can be used to automatically validate data as the program runs. However, since JSON Schema cannot contain arbitrary code, there are certain restrictions on the relationship between data elements that cannot be expressed. Therefore, any "validation tool" for a sufficiently complex data format may have two verification phases: one at the structural level and one at the semantic level. For structural level checks, you can use the Schema language. For semantic level checks, it may be necessary to use a more general programming language.

For the application, JSON Schema plays a very important role in the design and implementation of the API. An API implementation with JSON Schema validation capabilities can help with automatic validation of input data from users, as well as applications in a variety of scenarios that require validation when the program is running. This can help developers describe the JSON data hosted on their API in a more standardized and rigorous way, while writing less manual code.

For the data type and editing: In Json Schema we usually use the type keyword to agree on the data type. Corresponding to Json, the basic data types defined in Json Schema are as follows: string, Numeric types (integer, number), object, array, boolean, null. With the online editing tool (https://www.jsonschema.net/), it is easy to introduce the introduction of Json file, and automatically generate Json Schema, and then visually complete some of Json Schema's property definitions, greatly simplifying the writing workload.

5 Lecture 6

Question: Review and contrast the following three papers, found in the Files/graph-olap directory on Canvas:

1. Graph Cube: On Warehousing and OLAP Multidimensional Networks. Peixiang Zhao et al., 2011.
2. Entropy-based Selection of Graph Cuboids. Dritan Bleco and Yannis Kotidis, 2017.
3. Discovering interesting patterns in large graph cubes. Florian Demesmaeker et al., 2017.

Answer: Complex networks have been receiving increasing attention by the scientific community, thanks also to the increasing availability of real-world network data. In the last several years, the multidimensional nature of many real world networks has been pointed out. multidimensional attributes are associated with network entities formed the so-called multidimensional networks. Data warehouses and OLAP (Online Analytical Processing) technology have proven to be eective tools for decision support on relational data. However, they are not well-equipped to handle the new yet important multidimensional networks. In the paper Graph Cube: On Warehousing and OLAP Multidimensional Networks, the author introduces Graph Cube, a new data warehousing model that supports OLAP queries eectively on large multidimensional networks. By taking account of both attribute aggregation and structure summarization of the networks, they can aggregate network within every possible multidimensional space. Besides traditional cuboid queries, a new class of OLAP queries, crossboid, is introduced that is uniquely useful in multidimensional networks, which broke the boundary of the traditional OLAP model by straddling multiple cuboids within one query.

A graph cube represents all the possible aggregations of the initial multidimensional network, Graph cube mining is then no longer limited to the initial graph, but also explores all possible aggregations of the graph. the graph cube contains an exponential number of aggregate networks, In a big data setting, an analyst is overwhelmed by the number of patterns. Hence, two methods to help the analyst quickly locate interesting relationships in the aggregated graphs were proposed.

The first method proposed by the paper Entropy-based Selection of Graph Cuboids uses information entropy. The proposed entropy-based filtering of data reveals irregularities and nonuniformity, which are often what the decision maker is looking for. Basically, they use the information entropy to elevate parts of the graph cube that experience disorder. the entropy-based filtering prunes significant parts of the graph cube and at the same time gives valuable indicators as to where interesting associations exist.

The second method proposed by the paper Discovering Interesting Patterns in Large Graph Cubes focus on the mining of associations between entity features in networks. the authors proposed a hypothesis testing framework to evaluate how surprising the found patterns are with respect to an independence model. They also showed the relationship between their pattern mining framework and the frequent itemset mining literature, In addition, they proposed a mapping from attributed graphs to transactional databases and did some comparison with the frequent itemset mining algorithm MINI and their method.

To sum up, the paper Graph Cube: On Warehousing and OLAP Multidimensional Networks proposed a new data warehousing model, Graph Cube, which was designed specically for efcient aggregation of large networks with multidimensional attributes. Since In a big data setting it contain an exponential collection of aggregated graphs, the other two papers proposed two methods using information entropy or a hypothesis testing framework to quickly locate interesting relationships in the aggregated graphs.