Big Data Analytics

Mitschrift von Aaron Winziers

SS 2020 - Coronasemester

1 Introduction

1.1 3 Big Vs

- Volume
- Velocity - Data should be updated much more quickly - no longer work in batches
- Variety - Videos, text, from web etc

Veracity joins the other 3 Vs nowadays

1.2 Volume

- Average company has 100 TB of data
- 2.5 quintillion bytes created every day
- the amount of data created will be 300x greater in 2020 than 2005 (aggregate, estimate)

1.2.1 Challenges created by data volume

- Efficient storage
- Efficiant process queries
- Efficient learning with models
- What hardware and software architecture is needed for this?

1.3 Variety

- Data consists of different forms of data

1.3.1 Challenges created by data variety

- Syntactic heterogeneity - understadning different data types and formats
- Semantic heterogeneity - Differnt representations forthe same information
  Name abreviations - John Smith, J Smith, (Smith, John), Jon Smithe
- The prev 2 issues need to be understood because we need to combine:
  information from many different sources
  different types of information

1.4 Velocity

- The speed at which data is created and processed
- Data needs to be processed quickly or otherwise (sometimes) forgotten

1.4.1 Challenges created by data velocity

- Extremely fast flow of information
- Assessing the value of incoming information and drop "unimportant" information
- Quick integration of new information

1.5 Veracity

- Deals with the uncertainty of data
- Can you trust the data?

1.5.1 Challenges created by data velocity

- Differnet kinds of data defects:
  Data may be invalid (broken sensors, bad software)
  Data may be biased and not reflect the true population
  Data may be manipulated
- Methods are needed to identify and "repair" data defects

1.5.2 User-Generated Data

- Users may answer dishonestly or not take surveys seriously
- Users may try to purposely influence the results of surveys
- Must check the plausibility of the data before using

1.6 Real Life Scenarios

Twitter and facebook have hella data to process

1.6.1 Search engines

- Analysis of User behavior - related queries, useful books, etc.
- Result rankings need to be processed
- Voice query processing
- Question answering - not just returning web results
- Understanding images - Showing quick summarizing graphics
- Velocity - i.e. news needs to be current

1.6.2 Online shops

- Further shopping suggestions
- Bundles that are often bought together
- Adjusting pricing
- Fraud detection - esp. in reviews

1.7 Data Warehouse s Data Lake

1.7.1 Data warehouse

- Data is processed into schema before being pput into warehouse
- Data is structured
- Analytics are then performed on clean data
- Many decisions need to made in advance esp when deciding which data to keep
- Poor approach with dynamic data or with multiple sources (No guaranteed schema)

1.7.2 Data Lake

- Unstructured data is gathered and stored
- To analyze, data is selected from data pool
- No decisions are made about what to keep
- ALL data interesting for analysis is kept - both self created and gathered
- All data stored in single system dedicated only to storing data

1.7.3 Modern Big Data environment

- All data fed into data lake
- Regular analysis is sent to data warehouse

  Used for established mining processes

  Extract, transform, load
- Analytic sandbox

  more exploratory analysis

  Used for more flexibility

1.8 Web scale computation

1.8.1 Why is volume an issue?

- Reading data from disks is slow - esp when only from one disk
- Reading must be performed in parallel
- Larger amounts of data are more difficult to process
- Web - scale computation
  - Web crawlers gather large amounts of data (commoncrawl.org 220TB)
  - Required analyses:

    Document inversion - creating a search index

    PageRank

    Web log mining (identifying user behavior)

    Trend Mining - predicting upcoming topics

1.8.2 Scaling computing power (Data centers)

- Buying many cheap computers often cheaper than buying more powerful computers
- Buying more = scaling out
- Buying more powerful = scaling up
- Issues with scaling out

  Large number of machines -¿ many hardware failures, esp hard drive failures

  Distributed solutions and algorithms are required

1.9 Fallacies

1.9.1 Hardware failures

- Failures are common, not the exception
- With larger amounts of machines the probability that something will fail approaches 1

1.9.2 Fallacies of Distributed Computing

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

1.9.3 Failure handling

- Failures happen at any time - needs to be compensated by algorithms
- Data can be replicated - in Hadoop in 3 locations
- The state of the information can be logged
- Tasks can be performed redundantly

1.10 Hadoop

- De facto standard for web scale analytics
- Open source software for reliable and scalable distributed computing
- Uses simple programming models - code does not change between one and many machines

2 Data Quality

2.1 Intro

- Garbage produces garbage
- A model trained with bad data will produce bad results even with good data

2.1.1 Dimensions of data quality

Completeness

- All expected data is available
- A sufficient amount of data is available
- Completeness is dependent on application
  Data insufficient for one question may still be used to answer another
- No objective measure

Accuracy

Clear

Currency

- Is the data new enough for my application?
- Is new data added fast enough (goes to completeness)

Consistency

- Conforms to an authority(isbn) **(or)**
- Does ot contradict itself
- Hard inconsistency:
  Binary decision(consistent or not)
  Relatively simple

- Soft inconsistency:

  Value looks suspicious

  May or may not be correct

  Often related to other quality dimensions

2.1.2 Differences to Classic data

- More heterogeneous (More variety & from different sources)
- Changes faster (Velocity)
- Cannot reject bad data (goal is to gather as much as possible)

  Goal is to gather as much as possible

  Different projects/applications need access to the data and have different ideas to the quality of the data

2.2 "Good old days"

- Define data constraints
- Check if data meets constraints
- Reject otherwise

2.2.1 How to check consistency

Data definitions:

- SQL constraints
- DTD/XML-Schema

Authorities:

- Standards (ISO 639, ISO 3166-2)
- Authority files (e.g. Placenames)

2.2.2 Constraints

- Rules may become very generic (* is NOT a good constraint)
- Can become very specific (impossible to maintain)
- They were bad for classical data, horrible for big data

  Big data is even more heterogeneous and much more volatile

In big data

- Constant need for updating
- What happens with unexpected data?

Combining data pools

- Old approach meant migrating and rewriting old data
  Not feasible because of cost of changing and the speed at which rules change
  Each data provider has different rules and standards

2.2.3 Rejecting data

- Data can no longer be rejected
- We don't know what is wrong
- We don't have time to fix it
- We may need the "bad" data later

2.2.4 What to do

- Have a clear understanding of what good data is
- Filter bad data based on your current application
- Critically check your results
- Data lake =¿ Filter =¿ Processing =¿ Results (can jump back in every step)
- New pipeline for every application

2.3 Steps to answering a question

1. What are the data sets?
2. Filtering
   - What is the expected value range?
     Most often times must be answered by a domain expert
   - Why is information missing?
     Answer also depends on domain knowledge. There may be no answer.
   - Is it okay that information is missing?
     Domain expert
   - How much data is missing?
     Dunno

2.3.1 Know your data

- Outlier detection / analysis
- Descriptive statistics
- Result visualization

Outliers

- Not all outliers are bad
- Decision can be made to accept the loss of good data through cutoff (like in dates)
- Statistical analysis can show problematic outliers and systematic problems
- It **cannot** find individual errors
- Processing affects answerable questions

Do not filter the wrong data Keep your bias out of the pipeline

3 Data Quality Part II - Author Name Disambiguation

3.1 Problem

- Direct references - multiple entities with same name
- Indirect references - Generally difficult - sensitive to context(time of quote i.e. Obama or Trump), sensitive to language (Monaco, Munich, München)

3.2 Author name Disambiguation vs Named Entity Disambiguation

- AND is a simplification
- Always deals with persons (sometimes groups)
- Mentions are already extracted from text
- Information is already *relatively* structured
- Goal is to have minimal homonyms and synonyms

3.3 Problems with using names

- Names change

    Change in cultural context - change in translation o.Ä.

    Marriage

    Visit Mecca - add Haj

3.4 How to identify a person

- Use other properties:
  - Coauthors
  - Affiliations (Uni, research group etc)
  - Topic (Research specialization)

3.4.1 However:

- Coauthors may also be synonyms/homonyms
- Data may be difficult to interpret - Universität vs Univerisity of
- Data may just be wrong

3.4.2 Also possible:

Use external IDs:

- Authority IDs (National Libraries)
- Specialized IDs (Orchid)
- Third Party (email, twitter handle etc)

3.5 Algorithmic solutions

Two principal solutions:

- Batch computations: compute a matching for all mentions at once
- Iterative approach: Add new mentions to the existing profiles

3.5.1 Batch Computation

1. Extract features for each mention (Coauthors, affiliations, etc.)
2. Calculate similarity value for all pairs of mentions $n^2/2$ values (very expensive)
   - Cosine similarity
   - Trained similarity

3. Clustering to create individual profiles
   - Very low values can be ignored (don't need to evaluate the whole graph)
   - Provides a bit of an issue
   - Requires a lot of data

Pros and Cons Pro:

- Ignores previous errors
- Treats new data and old data equally

Cons:

- Slow
- Cannot run with every update
- Mentions can oscillate between entities

Blocking

- Makes matching faster
- Divides mentions by a simple rule (last name)
- Matching is run per block

Pro:

- Simple to implement
- Improves speed (reduces required similarity values)

Cons:

- Entities may land in different blocks -¿ never matched
- Introduces structural problems -¿ changed names will not match

3.5.2 Iterative Approach

- Keep known profiles
- Search for best matching profile (or create new ones)
- May update existing profiles (combine previously separate profiles)

Pro:

- Can reuse known data (That was expensive to compute)
- More reactive for new data
- Compares similarities to entire profile (not individual publications)

Cons:

- Might still require blocking
- Propagates old errors
- Need to store profiles
- Needs to be able to correct profiles

3.6 Evaluation

2 Goals:

- Determine if matching is acceptable
- Find optimal configuration for algorithm (secondary but important)

Typical approach

1. Get a test collection
2. Run Algorithm
3. Analyze results

Test collections consist of:

- A list of mentions
- A predefined mapping between mentions and entities

Algorithm is good if it approaches the gold standard (defined in test collection)

Creating test collections is difficult:

- Use a predefined one
- Make your own
- Try to reuse data you already have

Predefined

- Make sure it matches your collections
- The features should be the same
- The features should carry the same weight
- Datasets may be unreliable

Self made

- All the same issues as with predefined
- Need to be created (expensive)
- Need to prevent bias in creation (May forget an edge case)
- May not be transferable (Need to create multiple)

Reuse own data Pros:

- Very cheap
- Covers wide array of cases

Cons:

- Only works if data is correct
- Limited in scope (Wei Wang -¿ homonym)

3.6.1 Evaluation

Quality of results

- Standard methods (precision, recall, cluster alignment)
- Method must match application

B^3 is an example algorithm

Performance of algorithms

- How long does it take?
- What resources are needed?
- Is your collection large enough?

4 Hadoop Framework

4.1 Introduction

- Open source big data processing framework
- Focused on processing large data sets on clusters of computers

4.2 Related Apache projects

Get from VL Folien 3-6

4.3 Distributed file system

- Idea is to move the computation to the data
  Aboids need for copying over network (slow)
- Files stored in large chunks in cluster
- **Namenode** tracks block location
- Computations should be performed on the node where the data is

4.4 File storage and replication

HDFS

- Files are separated into blocks
- Blocks are units for computations (blocks are always processed as a whole)
- Default replicas = 3

- Replication occurs when blocks are filled -¿ goes in to replication pipeline

  Block is full -¿ contact NameNode -¿ send data to other DataNode -¿ DataNode forwards to third node

4.5 MapReduce

4.5.1 Key idea

- Spread task of processing data
- According to map and reduce rules/functions
- Framework deals with node failures, load balancing etc

4.5.2 Map phase

- Machines process the block that they contain
- Produce key-value pairs which are used to partition the data
- 1 map task per data block
- Applying function f not influenced by other computations

  Allows for parallel computing
- Order in which computations are performed also unimportant

4.5.3 Reduce phase

- Data is aggregated for each key-group
- Should apply another function f and then an accumulator

4.5.4 Distributed MapReduce

- Input data is already chunked -¿ easy to process with single map task
- Some data blocks to not fit uniformly into blocks -¿ some data will need to be read across boundaries

Key principle

- Apply map function on each of input splits in dedicated process (run in YARN Container)
- One function call / line in input split (at least for text data)
- Produces data with keys -¿ partitions are based on keys

Computation

- Master node coordinates computation
  Accepts job(task)
  Computes map and reduce tasks
  Selects and activates worker nodes
- Worker node
  For map: selected if close to data
  Consumes intermediate results and creates final output

Parallelization

- Map functions can run in parallel
- Reduce functions can run in parallel
- Bottleneck: Reduce cannot start before map is finished