## Lecture 1 – 22/4

Learning about building tools to manage huge volumes of data.

Small parts of the book for literature

Slides aren’t study literature, only the book/article literature. (no previous exams)

Test is multiple choice and short answers, but mostly focused on project (60%)

Project takes a month (2 a 3 persons)

* Finding similar items (NN)
* NoSQL
* Key-Value Stores (Redis, ryak)
* Column stores (Cassandra, Hbase)
* Document Stores (Mongo DB, CouchDB)
* Graph alogirhmt & Neo4J
* Graph Database, Neo4J

## Lecture 2 – 24/4: Entity resolution (finding similar items)

Entity resolution: Decide whether 2 data structures correspond to the same real world entity.

* **Text variants**: Misspellings, acronyms, transformations, abbreviations
* **Local Knowledge**: Local use of formats
* **Data evolution**: Alternative/updated name appearing over time
* **New Data Functionality**: Other Database has conflicting/different info than the other (newer has different address which could mean someone moved houses)

4 solutions to fix:

* Atomic similarity methods
  + **Edit Distance:** updating/removing/inserting parts of the word closeby
    - 1 point for each insertion + 1point for each gap + 1point for extention
  + **Jaro Similarity**: Give similarity match percentage between 2 words using common characters
    - Jason-wrinkler: Extension that gives higher weight to matching prefix
  + **Soundex**: Match words with the same speaking tone



* **A diagram of a mathematical equation

  Description automatically generated**Similarity methods for sets
  + **Group linkage:** groups relational records (excl. unique records) with a number
  + **Merge-purge:** creates key for each related word combi and sort the relation using a standard format and only compare a limited set of relations.
  + **Prefix-based indexing:** If 2 strings have 90% similar, they must share at least 10% length of their string creating a base of index on symbol for the [x+1] pos

A black and white text

Description automatically generatedA screenshot of a computer

Description automatically generated



* Facilitating inner relationships
* Methods in uncertain data

Axioms of distance measure d is a function from pairs of points to real numbers then:

* D(x,y) >= 0
* D(x,y) = 0 iff x = y
* D(x,y) = D(y,x)
* D(x,y) <= D(x,z) + D(z,y) iff triangle

## Lecture 3 – 29/4

Hash function from A to B: Place numbers from one place (A) to multiple buckets (B) using model to reduce the scan time to find a number (you know which bucket it is in). This will reduce the complexity of searching for similar items.

When too many data are distributed between the ram and disk, it is called **thrashing**.

Given:

* A set of high dimensional data points (like an integer age)
* And some distance function: Which quantifies the “distance” between data points

Goal:

* Find all pairs of data points that are within some distance threshold

Applications:

* Pages with similar words (for classification and plagiarism)
* Netflix (similar tastes, users)

**K-gram**/**shingle** is a method to find similar documents. Which uses a sequence of characters where k is the amount of letters it looks forward (k=2 means it uses itself plus the next).

E.g.: you want to find the similar document z of [abcabfef] given document x [abcab] and document y [abfef]. Document z gives with 2-shingles {ab bc ca bf fe ef}, document x gives {ab, bc, ca} and document y {ab bf fe ef}. Using the preset of doc z, the other docs will be compared which documents has the most similar shingles using an integer array with common index of doc z.

**Jacardi similarity does the same, but using interactions**

Example: C1 = 10111; C2 = 10011 (not counting zeros)

* Size of intersection = 3; size of union = 4,
* Jaccard similarity (not distance) = 3/4
* Distance: d(C1,C2) = 1 – (Jaccard similarity) = ¼

A white paper with black text and numbers

Description automatically generatedBut for a million documents it takes 5 days…



Using **minhashing** you can convert large sets to short signatures, while preserving the similarity.

A diagram of a diagram

Description automatically generatedCheck for every document (x-as), create multiple random orders (3 in example) and go through each document (y-as starting at arrow) using a random order and check when for each document it is a 1. Place the index from the random order when the first 1 is found in the signature matrix M.



The Jaccard similarity probability is the same still with a/(a+b+c).

Still it requires to create multiple (only 3 in example) orders, which causes a lot of memory. So: Row hashing.

A screenshot of a math problem

Description automatically generated



## Lecture 4 – 5/1

B+Tree sort: Sort every value by checking if it higher/lower than the given value and by going through the branches you get to smaller gaps.

Binary search: For finding a single value, by sorting the data before (once) and taking the middle of the part the solution can be, it improves the complexity from O(N) to O(log(N)).

Split Trimming: For trimming the sequences to smaller parts, use multiple workers to split the data into multiple parts and let them trim sequentially, which improves the complexity from O(N) to O(N/k)

Map Reduce: For counting the sequences across multiple documents, use multiple workers for each document to map the occurrences to dataset x. Reduce x by counting every occurrence within the dataset and adding to a dataset y to the size with only the unique values.

A white hexagon with black text

Description automatically generated

Shuffle/Emit: Takes all key/values pairs and combines the values with the same keys

## Lecture 5 – 6/5 Map Reduce part 2

Inverted Map Reduce Index:

* 1, (pancakes for breakfast)
* 2, (dislike pancakes)
* 3, (eat breakfast)
* 4, (love eat)
* ((“pancakes”, (1,2)), (“breakfast”, (1,3)),( ‘eat’ (3,4))

Relation joining: join 2 databases with one similar column

1. Get every possible key value pair where the key is the similar column (ID) and the value is row data from either database x or database y (if x has 5 rows and y has 3 then 8 rows)
2. Then shuffle/emit it by creating a nested tuple for each unique similar column (ID).
3. Reduce it by combining for each key the values depending on the information they provide (department + employee but not department !+ department)

A diagram of a function

Description automatically generatedA blue arrow pointing to a number

Description automatically generated



### Map reduce implementation

~~

Project:

* Network server provider and you are an employer.
* Given servers with book buying information and credit card requests
  + Credit card information with mastercard or visa card
  + Also buys another book with a different card method (ideal)
  + Logs of every timestamp action (request and response)
  + Problem: Identify common patterns (like time) within these buying

## Lecture 6 – 13/5

**One learning window update method:** adding a new value to model and learning rate (concept drift problem)

**Sampling data stream method**: (randomly if full) store a reservoir size amount of elements n (elements seen so far) of the stream and discard the rest. If full (in this case n=6 with p= (n-reservoir size) /n=5/6) it is:

A diagram of a mathematical equation

Description automatically generated

Windowing models: Seeing a window of length N (N recent elements) of the data stream

* Sliding: Shift one element at a time
* Tumbling/Disjoint: Shift N amount of elements
* Hopping: Shift S amount of elements at a time
* Exponentionally decaying windows
  + Give every sample an importance weight (recent higher than old)

Data stream querying examples

* Filtering by selecting elements with a property (email)
* Counting distinct by using last k elements (supermarket sales)
* Estimating by using last k elements to estimated the avg./std.
* Frequent finding (movies)

### Filtering

Given 1M email addresses and 8M bits to filter. The probability to get a wrong address is p=1/8.

Using throwing darts the percentage of missing is 1 – e^(-M/N) = 1- e^(-1/8) = 0.1175

Bloom filter formula:

* 𝑀 is the number of bits in the Bloom filter (size of ∣𝐵∣)
* 𝑁 is the number of elements in the set ∣𝑆∣
* D is the number of hash functions

To reduce the False positive you can:

* Increase bloom filter size
* Decrease hash functions
* Adjust parameters
* Different filters

!TODO check how to do the calculations

Flajolet-martin approach for counting zeros that are connected to the last e.g:

01010 (1)

10000 (4)

01000 (3)

11010 (1)

Highest is 4 so using the estimated number of distinct elements = 2^R = 2^(4-1) = 8