Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Large-Scale Duplicate Detection

Potsdam, April 08, 2013

Felix Naumann, Arvid Heise

Outline

Overview over Freedb

- Provides metadata about CDs
  - Artist, title, tracks, times, category
  - More or less equivalent to ID3v1
- Acquired by MAGIX in 2006
- 6.6m requests in 2012 (declining)
- Used by many CD players/rippers and Mp3 taggers

Quality in Freedb

- Very complete, but many duplicates

Words:	chocolate starfish	OK
Search:	○ **All**	◉ **Select** ☐ Artist ☑ Title ☐ Track ☐ Rest
Categories:	◉ **All**	○ **Select** ☐ Blues ☐ Classical ☐ Country ☐ Data ☐ Folk ☐ Jazz ☐ Misc ☐ New Age ☐ Reggae ☐ Rock ☐ Soundtrack
Grouping:	○ **By category**	◉ **One list**

Search results

684 result(s) found displayed on 69 page(s).

Limp Bizkit / Chocolate Starfish	**Details**
Limp Bizkit / Chocolate Starfish	**Details**
Limp Bizkit / Chocolate Starfish...	**Details**
Chocolate Starfish / Chocolate Starfish	**Details**
Chocolate Starfish / Chocolate Starfish	**Details**
Limp Bizkit / Chocolate Starfish	**Details**
Limp Bizkit / Chocolate Starfish ...	**Details**
Limp Bizkit / The Chocolate Starfish	**Details**

Quality in Freedb

Search results

684 result(s) found displayed on 69 page(s).

Limp Bizkit / Chocolate Starfish

Limp Bizkit / Chocolate Starfish

Limp Bizkit / Chocolate Starfish...

Chocolate Starfish / Chocolate Starfish

Chocolate Starfish / Chocolate Starfish

Limp Bizkit / Chocolate Starfish

Limp Bizkit / Chocolate Starfish ...

Duplicates in Freedb

Limp Bizkit / Chocolate Starfish	Details

Tracks: 15
Total time: 65:37
Year: 2000
Disc-ID: misc / d50f660f

1.	Intro	0:06
2.	Hot Dog	1:20
3.	My Generation	3:52
4.	Full Nelson	3:42
5.	My Way	4:09
6.	Rollin' Air Raid Vehicle	4:35
7.	Livin' It Up	3:35
8.	One	4:26
9.	Getcha Groove On	5:44
10.	Take A Look Around	4:31
11.	I'll Be Ok	5:21
12.	Boiler	5:08
13.	Hold On	7:00
14.	Rollin' Urban Assault Vehicle	5:45
15.	Outro	6:23

Limp Bizkit / Chocolate Starfish	Details

Tracks: 15
Total time: 75:52
Year: 2000
Disc-ID: rock / da11ce0f

1.	Intro	1:20
2.	Hot Dog	3:52
3.	My Generation	3:43
4.	Full Nelson	4:09
5.	My Way	4:37
6.	Rollin' (Air Raid Vehicle)	3:38
7.	Livin' It Up	4:26
8.	The One	5:45
9.	Getcha Groove On	4:50
10.	Take A Look Around	5:20
11.	It'll Be Ok	5:08
12.	Boiler	7:01
13.	Hold On	5:49
14.	Rollin' (Urban Assault Vehicle)	6:24
15.	Outro	9:50

Duplicates in Freedb #2

Limp Bizkit / Chocolate Starfish — **Details**

Tracks: 15
Total time: 75:52
Year: 2000
Disc-ID: rock / da11ce0f

1.	Intro	1:20
2.	Hot Dog	3:52
3.	My Generation	3:43
4.	Full Nelson	4:09
5.	My Way	4:37
6.	Rollin' (Air Raid Vehicle)	3:38
7.	Livin' It Up	4:26
8.	The One	5:45
9.	Getcha Groove On	4:50
10.	Take A Look Around	5:20
11.	It'll Be Ok	5:08
12.	Boiler	7:01
13.	Hold On	5:49
14.	Rollin' (Urban Assault Vehicle)	6:24
15.	Outro	9:50

Limp Bizkit / Chocolate Starfish... — **Details**

Tracks: 13
Total time: 63:56
Disc-ID: misc / a90f010d

1.	Hot Dog	3:52
2.	My Generation	3:41
3.	Full Nelson	4:08
4.	My Way	4:34
5.	Rollin'	3:35
6.	Livin' it Up	4:25
7.	The One	5:41
8.	Gatcha Groove On	4:29
9.	Take a Look Around	5:19
10.	It'll Be OK	5:07
11.	Boiler	6:58
12.	Hold On	5:45
13.	Rollin' [RMX]	6:22

Outline

1. Freedb
2. Seminar Overview
3. Duplicate Detection
4. Map-Reduce
5. Stratosphere
6. Paper Presentation
7. Organizational

Goals

- Learn about duplicate detection
- Work with large data
- Study related work independently
- Implement efficient and scalable algorithms
- Evaluate and incrementally improve algorithms

Seminar Mode

- 6 LP project seminar
- Mostly consultations instead of group meetings
- 4 teams with 2 students each
- **Students who participate in a master course about Hadoop/Stratosphere at our chair come last**

Grading

- Two short and one long presentations
- Implementation (strategies)
- Final report with evaluation
- Participation in discussions/consultations

Outline

1. Freedb
2. Seminar Overview
3. Duplicate Detection
4. Map-Reduce
5. Stratosphere
6. Paper Presentation
7. Organizational

Duplicate Detection

- Naïve: find all pairs of CD
  - Label pair as (non-)duplicate

Duplicate Detection

- Naïve: find all pairs of CD
  - Label pair as (non-)duplicate
- Problem: When are they duplicates?
  $\rightarrow$ similarity measure

Duplicate Detection

- Naïve: find all pairs of CD
  - Label pair as (non-)duplicate
- Problem: When are they duplicates?
  $\rightarrow$ similarity measure
- Problem: $\frac{2m*(2m-1)}{2} =$ too many comparisons
  $\rightarrow$ candidate selection

Similarity Measure

- Usually composition of smaller similarity measures

Similarity Measure

- Usually composition of smaller similarity measures
- Levenshtein (Edit) distance: for small typos

Similarity Measure

- Usually composition of smaller similarity measures
- Levenshtein (Edit) distance: for small typos
- Numerical distance: for years

Similarity Measure

- Usually composition of smaller similarity measures

- Levenshtein (Edit) distance: for small typos
- Numerical distance: for years
- Jaro-Winkler distance: more emphasize on the beginning

Similarity Measure

- Usually composition of smaller similarity measures

- Levenshtein (Edit) distance: for small typos
- Numerical distance: for years
- Jaro-Winkler distance: more emphasize on the beginning
- Jaccard: set similarity (several words or tracks)

Similarity Measure

- Usually composition of smaller similarity measures

- Levenshtein (Edit) distance: for small typos
- Numerical distance: for years
- Jaro-Winkler distance: more emphasize on the beginning
- Jaccard: set similarity (several words or tracks)

- Similarities compute a value in [0; 1]
- Threshold for dividing duplicates from non-duplicates

Candidate Selection

- Reduces the search space by sacrificing some recall

Candidate Selection

- Reduces the search space by sacrificing some recall
- Blocking: compare tuples that share a common value (prefix) in one attribute, e.g. genre or first word of artist

Candidate Selection

- Reduces the search space by sacrificing some recall
- Blocking: compare tuples that share a common value (prefix) in one attribute, e.g. genre or first word of artist
- Sorted Neighborhood: compare tuples that are near each other in a total order, e.g. title of album

Candidate Selection

- Reduces the search space by sacrificing some recall

- Blocking: compare tuples that share a common value (prefix) in one attribute, e.g. genre or first word of artist
- Sorted Neighborhood: compare tuples that are near each other in a total order, e.g. title of album
- Multiple passes: repeat with different attributes/orders, e.g. SNM of artist, then title

Candidate Selection

- Reduces the search space by sacrificing some recall
- Blocking: compare tuples that share a common value (prefix) in one attribute, e.g. genre or first word of artist
- Sorted Neighborhood: compare tuples that are near each other in a total order, e.g. title of album
- Multiple passes: repeat with different attributes/orders, e.g. SNM of artist, then title
- For SNM or multi-pass, use transitive closure
- If $a \simeq b \wedge b \simeq c \rightarrow a \simeq c$

Outline

1. Freedb
2. Seminar Overview
3. Duplicate Detection
4. Map-Reduce
5. Stratosphere
6. Paper Presentation
7. Organizational

Basic Idea

- Introduced by Google in 2004 [1]

- Usual program is 1 – 10 MB large
- We want to process GBs up to TBs of data
- Inefficient to bring data to program
- Solution: bring the program to the data
- Divide the program in map and reduce parts

- Works best for I/O-bound problems

Map, Reduce

- Map and reduce are *second-order functions* and origin in functional programming
- Take a set of `data` and a first-order function `func` as parameters

  ```
  secondOrderFunc(data, func)
  ```

- Apply `func` to tuples of `data`
- Example: `filter([1, 2, 3], { num -> num % 2 = 1 })`

  → `1`
  `3`

Key-Value Data Model

- Each tuple in `data` is a key-value pair `(k, v)`
- `k` and `v` may be arbitrary user-defined types
- Within one `data` set, `k` and `v` must be homogeneous
- `k` is comparable/sortable
- `k` and `v` must be serializable

Map

- Every key-value pair in `data` is processed independently:
  - Transfer `func` to all nodes to the cluster
  - Apply `func` to each local pair
  - No need to transfer any data over the network

- Definition: $map([(k_1, v_1), \ldots, (k_n, v_n)], func)$
  $\rightarrow [func(k_1, v_1), \ldots, func(k_n, v_n)]$

- Example:

```
udf(key, value) { return (key, value + "x") }
map([(1, "a"), (2, "b"), (1, "c")], &udf)

→ (1, "ax")
  (2, "bx")
  (1, "cx")
```

Reduce

- Groups all tuples with same key:
  - Globally partition data
  - Transfer `func` to all nodes to the cluster
  - Apply `func` to each partition
  - Might cause heavy network traffic

- Definition:
  $reduce([(k_1, v_1), (k_1, v_2), \ldots, (k_n, v_{m-1}), (k_n, v_m)], func)$
  $\rightarrow [func(k_1, [v_1, v_2, \ldots]), \ldots, func(k_n, [\ldots, v_{m-1}, v_m])]$

- Example:

```
udf(key, values) { return (key, concat(values)) }
reduce([(1, "a"), (2, "b"), (1, "c")], &udf)
→ (1, "ac")
  (2, "b")
```

Word Count

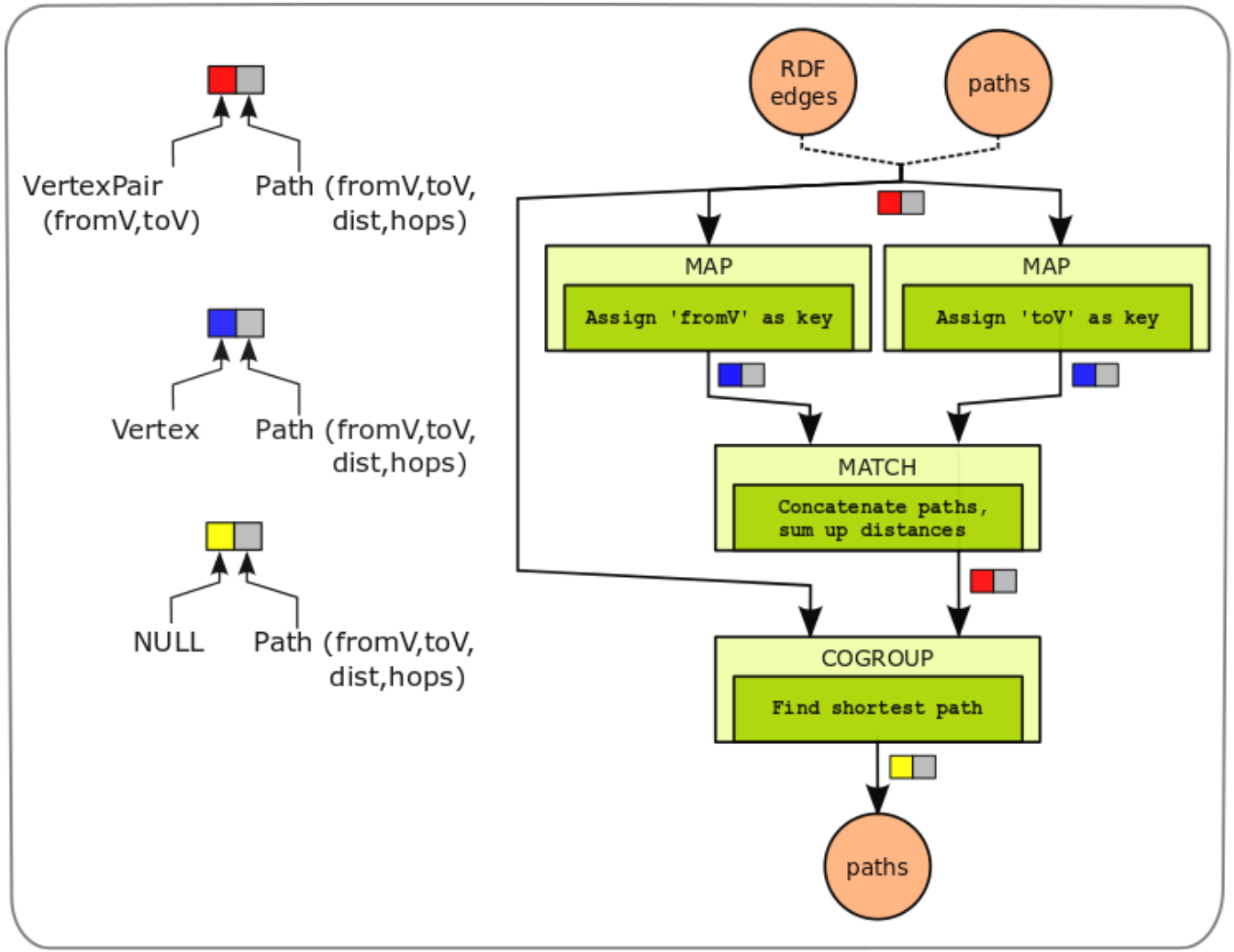
- Common example as it can be perfectly ported to Map-Reduce
- Two phases: tokenize sentences, count occurences
- Map UDF: split line into words, emit `(word, 1)`
- Reduce UDF: sum up `(word, 1), (word, 1), (word, 1)` $\rightarrow$ `(word, 3)`

Outline

1. Freedb
2. Seminar Overview
3. Duplicate Detection
4. Map-Reduce
5. Stratosphere
6. Paper Presentation
7. Organizational

- Research project by HU, TU, and HPI
- Improve short-comings of M/R for complex data flows
- Programs are specified as acyclic directed graphs
- Additional second-order functions with two inputs
- Robust and adaptive query optimization
- Exploit elasticity of clouds (automatically book/unbook VMs)
- (Intelligent fault tolerance)

Stratosphere Plan



Outline

1. Freedb
2. Seminar Overview
3. Duplicate Detection
4. Map-Reduce
5. Stratosphere
6. Paper Presentation
7. Organizational

Paper Presentation

- First presentation = paper presentation
- Fast knowledge transfer
- Different M/R algorithms that can be used in this seminar
- Each group presents a different paper = 4 of 6
- **After presentation, each group can choose any approach**

- Explain basic idea + M/R sketch in 15min
- Propose usage for our use case

Papers #1: Similarity Measures

- Efficient parallel set-similarity joins using MapReduce
  - Fast implementations of set similarities joins (Jaccard)

Papers #1: Similarity Measures

- Efficient parallel set-similarity joins using MapReduce
  - Fast implementations of set similarities joins (Jaccard)
- A Scalable MapReduce Framework for All-Pair Similarity Joins of Multisets and Vectors
  - Even faster for a special subclass of set similarity joins

Papers #1: Similarity Measures

- Efficient parallel set-similarity joins using MapReduce
  - Fast implementations of set similarities joins (Jaccard)
- A Scalable MapReduce Framework for All-Pair Similarity Joins of Multisets and Vectors
  - Even faster for a special subclass of set similarity joins
- Fuzzy Joins Using MapReduce
  - Implements efficient edit distances

Papers #2: Candidate Selections

- Multi-pass Sorted Neighborhood Blocking with MapReduce
  - Implements SNM with load balancing on Hadoop

Papers #2: Candidate Selections

- Multi-pass Sorted Neighborhood Blocking with MapReduce
  - Implements SNM with load balancing on Hadoop
- A fast approach for parallel deduplication on multicore processors
  - Two tier hash blocking with load balancing

Papers #2: Candidate Selections

- Multi-pass Sorted Neighborhood Blocking with MapReduce
  - Implements SNM with load balancing on Hadoop
- A fast approach for parallel deduplication on multicore processors
  - Two tier hash blocking with load balancing
- CBLOCK: An Automatic Blocking Mechanism for Large-Scale De-duplication Tasks
  - Semi-automatic blocking; possibly too general
  - Contains interesting algorithms for the seminar

Outline

Dates

- April 08: Topic introduction
- April 12: **Application with paper wishlist**
- April 13: Notification
- April 15: Lecture: M/R and Stratosphere (open for everyone)
- May 06: Paper **presentation** (15+5 min)
- June ??: Intermediate **presentation** (15+5 min)
- July ??: Final **presentation** (30+10 min)
- End of August: Final **report** (6-8 pages)

Consultation

- Every team gets a 30 min slot per week
- May be canceled (>1 day in advance)
- Mandatory in the first two weeks to discuss the paper
- Mandatory one week before each presentation to discuss slides
- Mandatory in July/August to discuss paper outline

Infrastructure

- Common infrastructure
  - Mailing list
  - Common repository
  - Trac/Wiki?
  - Cluster with ten nodes (access and time schedule tbd)
- Individual infrastructure
  - Linux (Ubuntu) (VM) recommended for easy installation
  - Mac works, Windows does not work well
  - Stratosphere local testing
  - Unit tests for map/reduce tasks

Competition

- We provide two datasets of Freedb
  - One will be given to you in the beginning
  - The second is used to evaluate the performance
- The group that finds the duplicates fastest wins a small price
  - Has to meet a certain minimum quality
  - We provide a partial gold standard
- Only indirectly influences grade

Wish List

- Mail top 3 list to Arvid.Heise@hpi...
- Optionally add team partner
- If top 3 is identically with partner's wish list
  - One mail per team is enough
  - But add teammate in CC, so I can assume agreement
- If more than 8 students apply
  - Randomly select students with first wish for same paper
  - Fill in gaps with second wishes and so on
  - Team wishes will be honored as good as possible