Efficient parallel set-similarity joins using MapReduce

Speaker: Bibek Paudel
Tutor: Jörg Schad

January 28, 2011
Set Similarity

**Figure:** Set Similarity (Jaccard) is: $\frac{3}{8}$
Examples and Uses

- Detect Spam
Examples and Uses

▶ Detect Spam

▶ Detect mirrored web pages
Examples and Uses

- Detect Spam
- Detect mirrored web pages
- Detect plagiarism
Examples and Uses

- Detect Spam
- Detect mirrored web pages
- Detect plagiarism
- Information Extraction
Examples and Uses

- Detect Spam
- Detect mirrored web pages
- Detect plagiarism
- Information Extraction
- Distance between strings or documents
Different Metrics

- Edit Distance
- Hamming Distance
- Overlap coefficient
- Similarity measures
Different Metrics

- Edit Distance
- Hamming Distance
- Overlap coefficient
- Similarity measures

Figure: Sample duplicate records

---

*aAdaptive Name Matching in Information Integration, Bilenko et al, IEEE Computer Society*
Challenges

- Find similarity between all pairs?
- Find exact similarity or an approximation?
Challenges

- Find similarity between all pairs?
- Find exact similarity or an approximation?
- How to reduce the number of comparisons?
- How to use filtering?
Existing Methods

- length filter
Existing Methods

- length filter
- suffix and prefix filter
Existing Methods

- length filter
- suffix and prefix filter
- PPJoin [Example on board]
How to scale it up?

- Attractions of distributed system
How to scale it up?

- Attractions of distributed system
- MapReduce?
How to scale it up?

- Attractions of distributed system
- MapReduce?
- Working of MapReduce

**Figure**: MapReduce
The algorithm of the paper

Figure: Phase 1
The algorithm of the paper

**Figure**: Phase 2
The algorithm of the paper
Alternatives for each phase

- One phase token ordering
Alternatives for each phase

- One phase token ordering
- Kernel
Alternatives for each phase

- One phase token ordering
- Kernel
- One phase for phase 3
Alternatives for each phase

- One phase token ordering
- Kernel
- One phase for phase 3
- Total three M/R jobs
Alternatives for each phase

- One phase token ordering
- Kernel
- One phase for phase 3
- Total three M/R jobs
- R-S Join and Self-Join
Issues and shortcomings

- Dictionary size
- Candidates size
Issues and shortcomings

- Does it really scale up?
- Billions of pairs (depending on tokenization level)
- Experimental data set is too small to prove massive scale-up
Issues and shortcomings

- Does it really scale up?
- Billions of pairs (depending on tokenization level)
- Experimental data set is too small to prove massive scale-up
Possible Research Problems

▶ How to decrease the candidate blow-up?

▶ Storing the dictionary in some distributed key-value store?

▶ Exploiting the low number of candidates generated after map-phase?
Possible Research Problems

- How to decrease the candidate blow-up?
- Storing the dictionary in some distributed key-value store?
Possible Research Problems

▶ How to decrease the candidate blow-up?
▶ Storing the dictionary in some distributed key-value store?
▶ Exploiting the low number of candidates generated after map-phase?
Conclusion

- The problem of scale
- MapReduce is a nice paradigm for distributed large-scale jobs
- But we need specialized strategies
Questions?