Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Map Reduce on Hadoop
Seminar SS09

Similarity Join

Tim Felgentreff,
Andrina Mascher

Gliederung

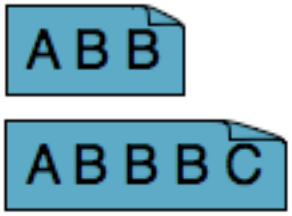
- Aufgabe
- Demo
- Algorithmus
- Performance
- Veränderte Aufgabenstellung:
  - Vergleich mit 1 Seite
- Ausblick
- Quellen

Aufgabe

- Aufgabe:
  - Similarity Join
  - d.h. ähnliche Objekte finden
- Umsetzung:
  - Wikipedia-Artikel untereinander vergleichen
  - Ähnlichkeit definiert über Jaccard

Aufgabe Jaccard

$$\frac{\text{gleiche Wörter}}{\text{Summe aller Wörter} - \text{gleiche Wörter}}$$



$$\frac{3}{8-3} = \frac{3}{5} = 0.6$$

Jaccard $J(A,B) = \frac{A \cap B}{A \cup B}$

Abwandlung auf Multimengen

DEMO

Algorithmus
3 Phasen

1. Pairing
   - Aus jeder Seite relevante Wörter finden
   - Wörter zählen
   - Für jede Gemeinsamkeit ein Seitenpaar bilden
2. Jaccard
   - Für jedes Seitenpaar

     erst jedes gemeinsame Wort betrachten

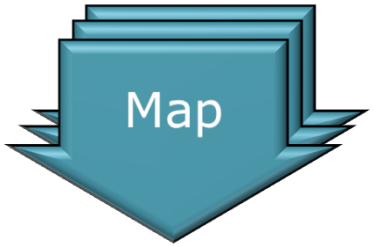
     dann alle Wörter eines Paares einrechnen
3. Output
   - Zu jeder Seite alle ähnlichen Seiten zuordnen

Algorithmus
Phase1 Map

page1 → ..test..head..head..

page3 → ..link..head..heading..

page2 → ..test..head..head..head..



für jedes Wort
- Relevanz prüfen (tf/idf)
- Wortstamm finden (Porter-Stemmer)
- Vorkommen zählen (Hash-Tabelle)

Gesamtwortzahl zählen

test → page1, 1,3
head → page1, 2,3

head → page3, 2,3
link → page3, 1,3

test → page2, 1,4
head → page2, 3,4

Algorithmus
Phase1 Reduce

head → page1, 2,3
→ page2, 3,4
→ page3, 2,3

test → page1, 1,3
→ page2, 1,4

link → page3, 1,3

- 2er-Kombinationen für Seiten finden
- Reihenfolge irrelevant, da Ähnlichkeit symmetrisch

page1, 2,3 → page2, 3,4
page2, 3,4 → page3, 2,3
page1, 2,3 → page3, 2,3

page1, 1,3 → page2, 1,4

∅

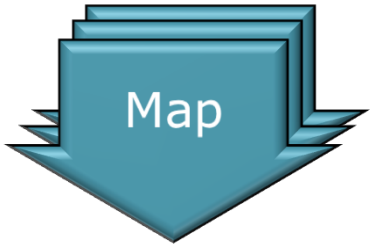
Algorithmus
Phase2 Map

page2, 3,4 → page3, 2,3

page1, 2,3 → page2, 3,4

page1, 1,3 → page2, 1,4

page1, 2,3 → page3, 2,3



für ein Seitenpaar errechne:
- Minimum beider Häufigkeiten für ∩
- Summe beider Wortanzahlen für ∪

page2, page3 → 2,7

page1, page2 → 2,7

page1, page2 → 1,7

page1, page3 → 2,6

Algorithmus
Phase2 Reduce

page1, page2 → 2,7
→ 1,7

page1, page3 → 2,6

page2, page3 → 2,7

für jedes Paar Jaccard berechnen:
- Wortvorkommen addieren =: S
- alle Vereinigungen wertgleich =: V

Paar ausgeben, wenn $\frac{S}{V-S} > k$

page1, page2 → 3/(7-3) = ¾

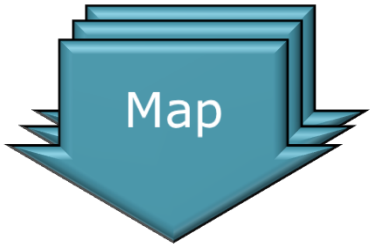
page1, page3 → 2/(6-2) = ½

page2, page3 → 2/(7-2) = 0.2

Algorithmus
Phase3 Map

page1, page2 → ¾

page1, page3 → ½



Relation symmetrisch

page1 → page2, ¾

page2 → page1, ¾

page1 → page3, ½

Seite3 → page1, ½

Algorithmus
Phase3 Reduce

page1 → page2, ¾
→ page3, ½

page2 → page1, ¾

page3 → page1, ½

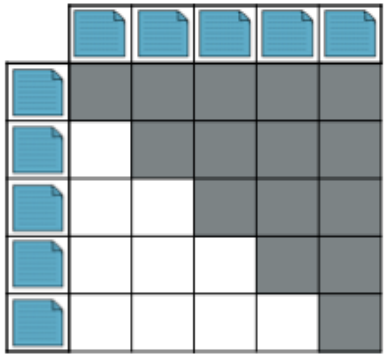
- Werte für eine Seite zusammenfassen
- eventuell Sortierung

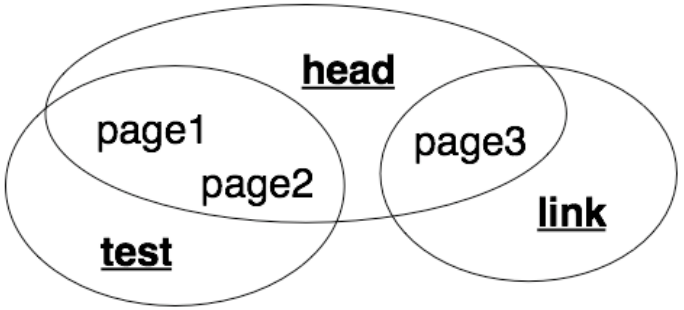
page1 → {page2 ¾,
page3 ½}

page2 → {page1 ¾}

Seite3 → {page1 ½}

Performance
Komplexität





Kreuzprodukt

Jaccard (Schnittmenge)

$O(n^2 * n^2)$

→ $O(n^4)$

Self Join

symmetrisch

$O(½\ n^4)$

→ $O(n^4)$

Wortcluster

$O(n^3)$

Performance
$O(n^3)$

P := Anzahl der Seiten W := Anzahl aller Wörter/Cluster

- Für jede Seite - $P * W$
  - relevante Wörter und Häufigkeit finden
- Jedes (Seite-Wort)-Tupel einem Cluster zuordnen - $P * W$
- Für jedes Wort - $W * P^2$
  - in jedem Seitenpaar rechnen
- Für jedes Seitenpaar - $P^2 * W$
  - alle beinhaltenden Cluster finden
  - dabei Jaccard berechnen

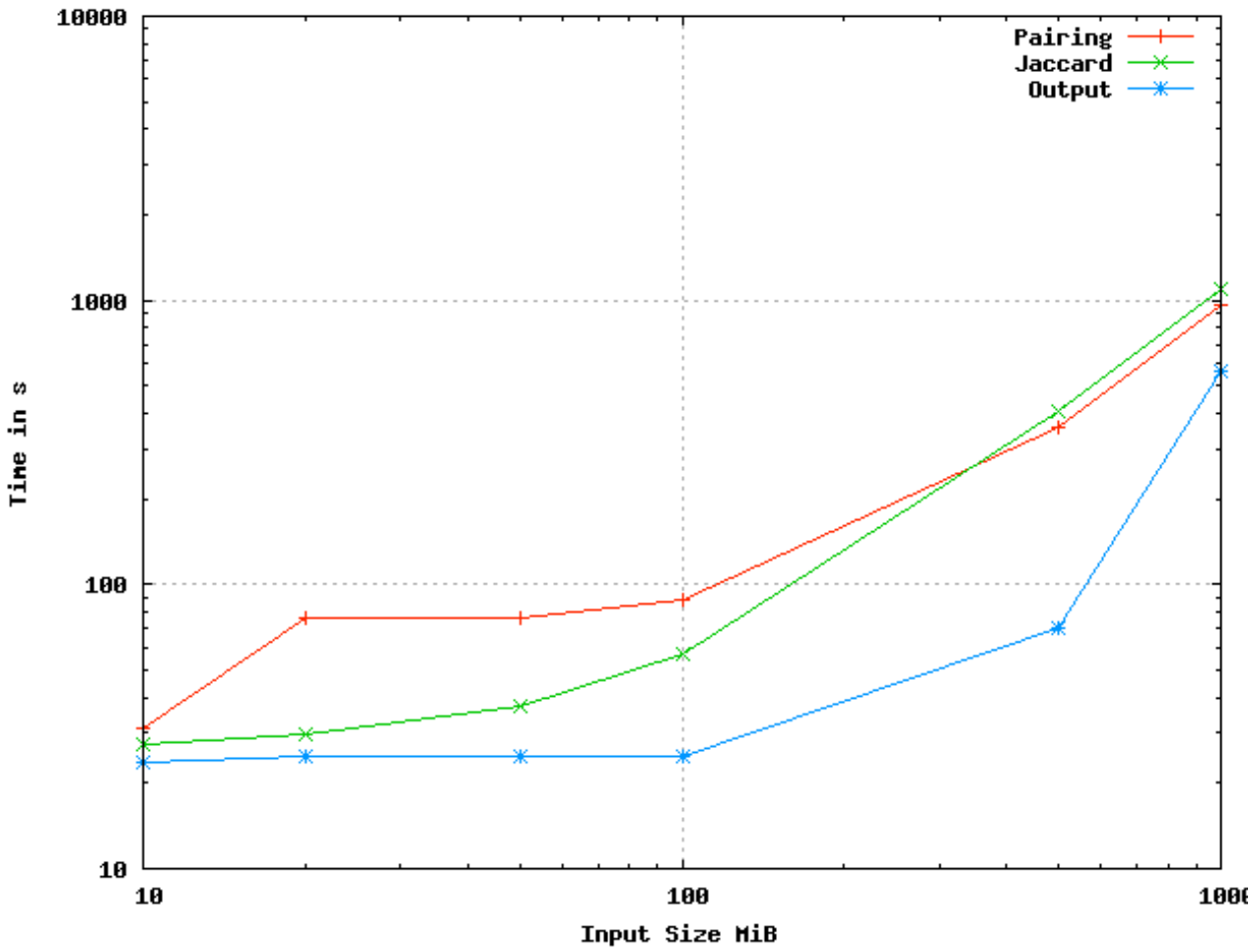
$2*P*W + 2*W*P^2$
→ $O(n^3)$

In der Praxis mit Map/Reduce weitere lineare Summanden

Performance

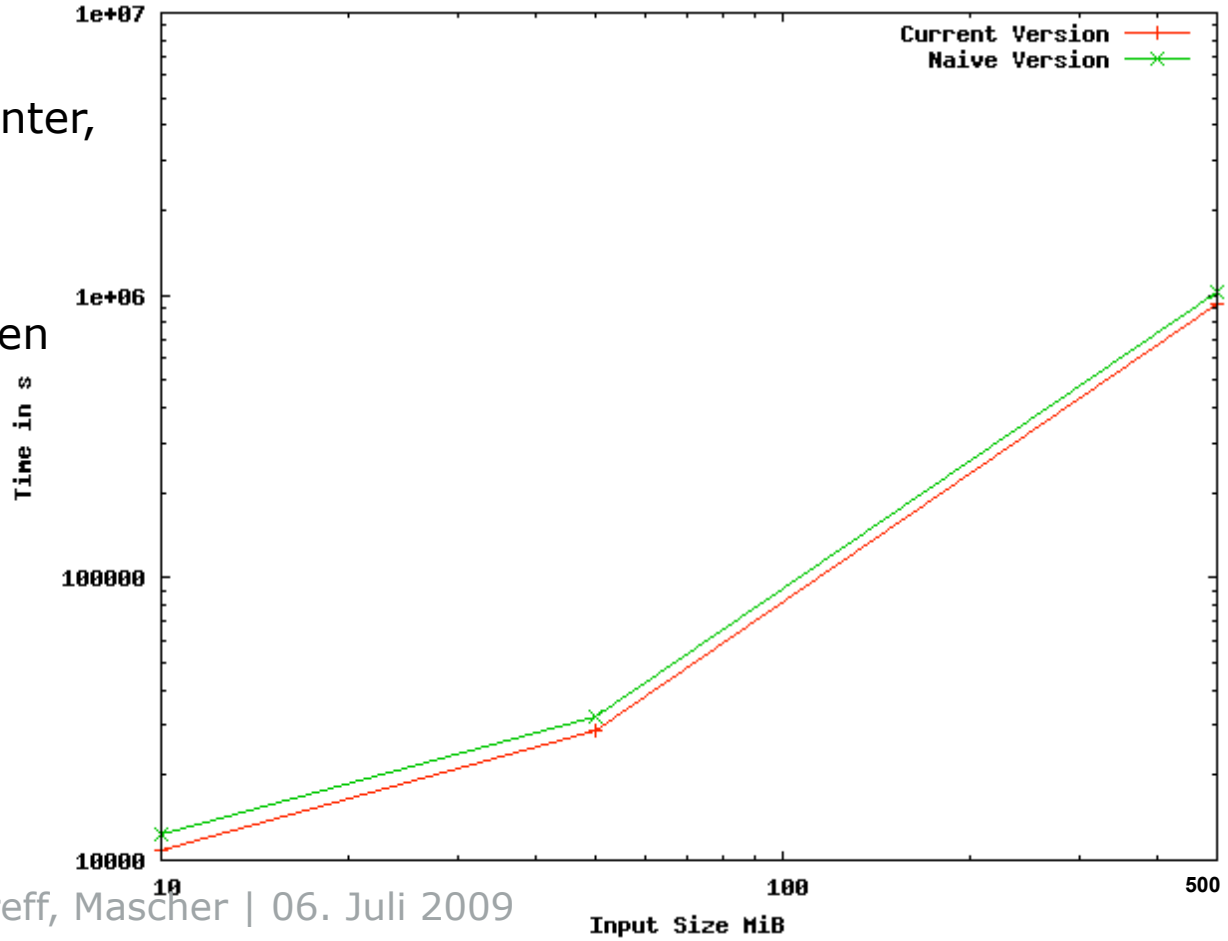
- Recht gleichmäßige Anstiege
- Bei großen Seitenzahlen werden mehr Paare gefunden

=> Jaccard und Output steigen stark an



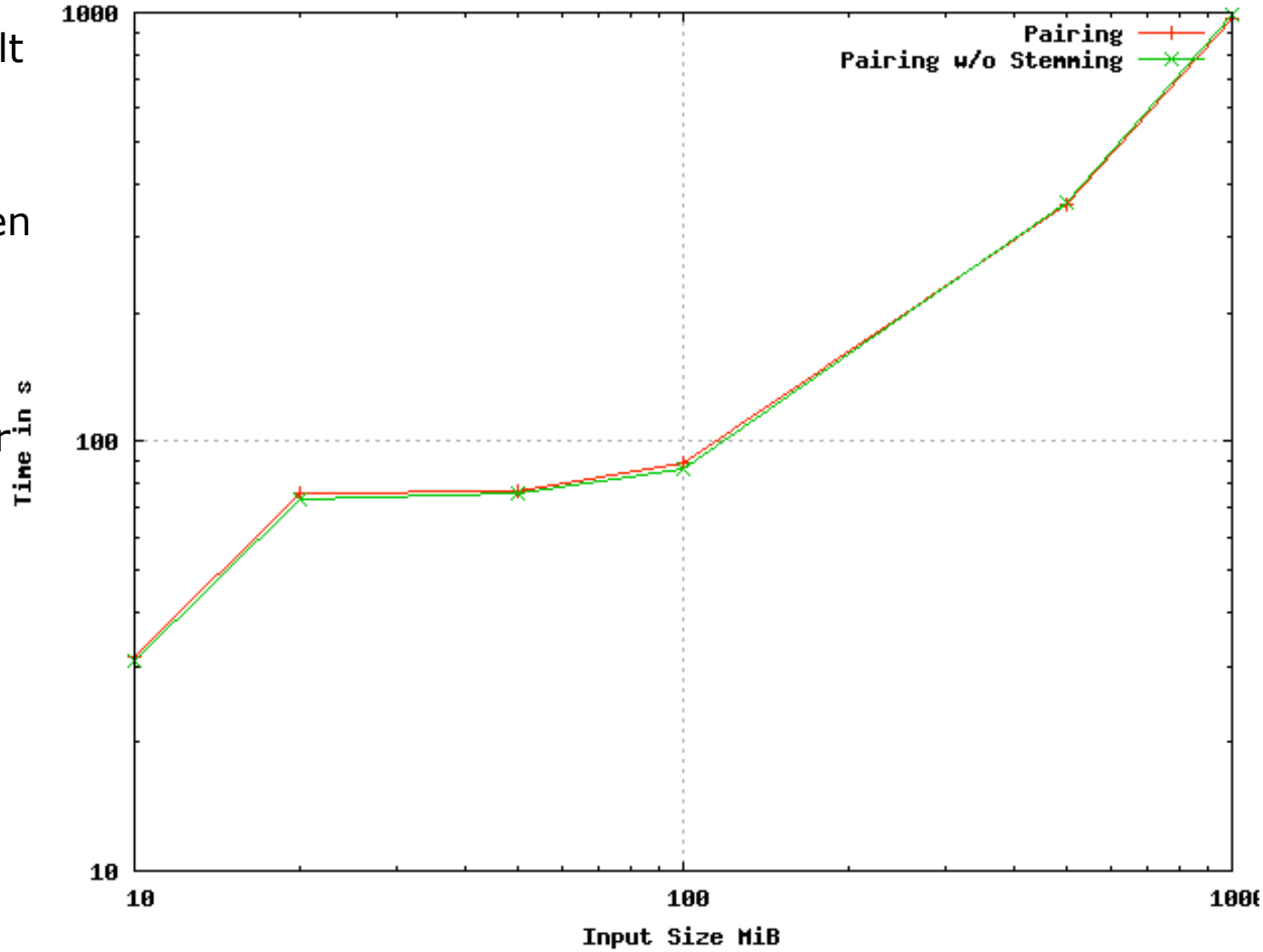
Performance
verschiedene Konfigurationen

- Erste Implementierung
  - Zum Zählen der Worte zunächst eine extra Wordcount Phase
  - Hashmap nicht wesentlich performanter, aber lesbarer
  - Übliche Optimierungen schinden die ein oder andere Sekunde

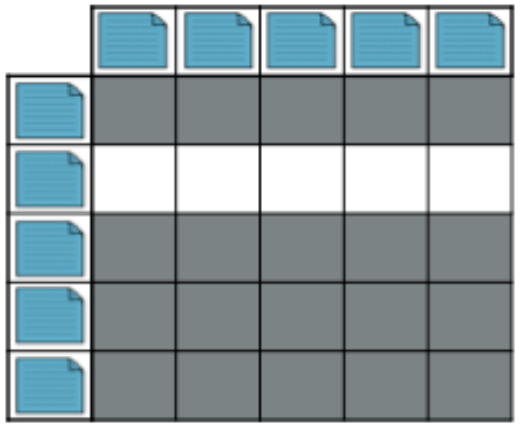


Performance - Word Stemming

- Stemming fällt kaum ins Gewicht
- Schwankungen liegen an Wortlängen -verteilung innerhalb der Seite



Veränderte Aufgabenstellung: Vergleich mit 1 Seite



- weniger Vergleiche
- aufwendige Abwandlung des Algorithmus
  - nicht effizient
  - Map/Reduce unangebracht
- Naïve Lösung besser:
  - Relevante Wörter aus Vergleichsseite suchen
  - Mit jeder anderen Seite vergleichen und Jaccard berechnen
  - simpel und minimaler Aufwand in $O(n^3+n)$

Ausblick

- relevante Wörter finden:
  - tf/idf
    - nicht Häufigkeit der Wörter zählen, sondern mit Relevanzmaß rechnen
  - zur Zeit einfach Links oder Überschriften
- mehrere Wortstämme extrahieren
  - bis auf n-Gram bisher keine hinreichend gute Lösung vorhanden
- Visualisierung der Ausgabe

Quellen

- Calculating the Jaccard Similarity Coefficient with Map Reduce for Entity Pairs in Wikipedia Jacob Bank, Benjamin Cole, Wikipedia Similarity Team (December 16, 2008)
- Porter Stemmer Algorithm in Java: http://www.ils.unc.edu/~keyeg/java/porter/PorterStemmer.java (12. Juni 2009)
- Developing a Word Stemming Program using Porter's Algorithm (Access Presentation), Venkatalakshmi K, Indian Institute of Science, 2001-2002
- Development of a Stemming Algorithm* by Julie Beth Lovins,† Electronic Systems Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (1968)
- A Comparison of Approaches to Large-Scale Data Analysis, Pavlo, Paulson, Rasin, Abadi, Dewitt, Madden, Stonebraker)
- http://en.wikipedia.org/wiki/Tf-idf (26. Mai 2009)
- http://hadoop.apache.org/ (25. Juni 2009)