

# Deduplikation

Daniel Stefan Heinz-Eugen Klose  
TU Dortmund University  
daniel-stefan.klose@udo.edu

## ABSTRACT

Ein wichtiger Teil des ETL-Prozesses ist die Deduplikation. In dieser Ausarbeitung werde ich mich genauer mit dem Thema der Deduplikation befassen. Es werden mehrere Metriken vorgestellt, an denen man Duplikate erkennen kann. Derweiteren werde ich eine Methode vorstellen mit der man diese effizient implementieren kann.

## PROBLEM

In einem OLTP-System sind die Daten meist "schmutzig". Das bedeutet, dass das ein Großteil der Daten auch nach wie vor mit der Hand von einzelnen Personen in das Datenbank-System eingetragen werden.

Bei diesen eingetragenen passieren verschiedene Fehler. Ein Fehler davon ist, das ein Eintrag falsch geschrieben ist bzw. mit verschiedenen Schreibweisen in dem Datenbank-System zu finden ist. Ein Beispiel ist dafür ist die Schreibweise für Dortmund. Ein Mitarbeiter eines Unternehmens macht unabsichtlich einen Schreibfehler und trägt Dortmund als Verkaufsort eines Produkts.

Dies führt dazu, dass nach der Überführung der Daten in ein Datawarehouse dieser Verkauf nicht zu den Verkäufen des Standortes Dortmund gehört und damit auch nicht bei den Aggregationen mit eingerechnet wird. Hier kommt die Deduplikation im ETL-Prozess ins Spiel.

## LÖSUNG

Das Finden und Zusammenführen von solchen Duplikaten wird im ETL-Prozess auch Deduplikation genannt. Bei der einfachsten Form für das finden von Duplikaten vergleichen wir, mit einem vorher festgelegtem Maß jedes Wort mit jedem anderen und führen diese zusammen welche unterhalb eines bestimmten Thresholds liegen.

Im weiteren Verlauf dieser Arbeit werden wir sehen, das dies eine sehr ineffiziente Lösung ist und schauen uns weitere Alternativen an.

## Vergleichsmaße

Als Erstes wollen wir uns verschiedene Vergleichsmaße ansehen, wie sie implementiert werden können, sowie ihre Vor- beziehungsweise Nachteile.

Das wohl einfachste Maß für die Gleichheit zweier Zwischenketten ist die sogenannte *Editierdistanz*, welche auch als *Levenshtein-Distanz* bezeichnet wird. Die Editierdistanz misst wie viele Operationen es braucht um den einen String in den anderen zu transformieren. Die Operationen lauten *erstzen*, wobei ein Zeichen durch ein weiteres ersetzt wird, *einfügen*, wobei ein Zeichen in einen String eingefügt wird und *löschen*, wobei ein Zeichen gelöscht wird. Um dies zu implementieren machen wir Gebrauch von dem Konzept der *dynamischen Programmierung*. Der Algorithmus ist in Pseudocode 1 zu sehen.

## Pseudocode 1 Editierdistanz mit dynamischer Programmierung

```
1: procedure DISTANCE( $u, v$ )
2:    $m := |u|$ 
3:    $n := |v|$ 
4:   decalre  $d[0..m, 0..n]$ 
5:   for  $i$  from 0 to  $m$  do
6:      $d[i, 0] := i$ 
7:   for  $i$  from 0 to  $n$  do
8:      $d[0, i] := i$ 
9:   for  $i$  from 1 to  $m$  do
10:    for  $j$  from 1 to  $n$  do
11:      if  $u[i] = v[j]$  then
12:         $cost := 0$ 
13:      else
14:         $cost := 1$ 
15:       $d[i, j] := \min($ 
         $d[i - 1, j] + 1,$ 
         $d[i, j - 1] + 1$ 
         $d[i - 1, j - 1] + cost)$ 
return  $d[m, n]$ 
```

Als weiteres Maß für das finden von Duplikaten ist die *Jaccard Similarity*. Diese misst die Ähnlichkeit von Mengen. Da wir aber keine Mengen vergleich sondern Strings, machen wir einen kleinen Trick und überführen unsere Strings in Mengen, indem wir jeweils zwei aufeinanderfolgende Zeichen des Strings zu einem Element der Menge machen. Zum Beispiel wird aus "Sweet" dann {"Sw", "we", "ee", "et"}. Damit können wir auch Strings vergleichen. Das Verleichen selber geschieht mit der Formel:

$$\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Als Beispiel vergleich wir *Sweet* und *Sweat*.

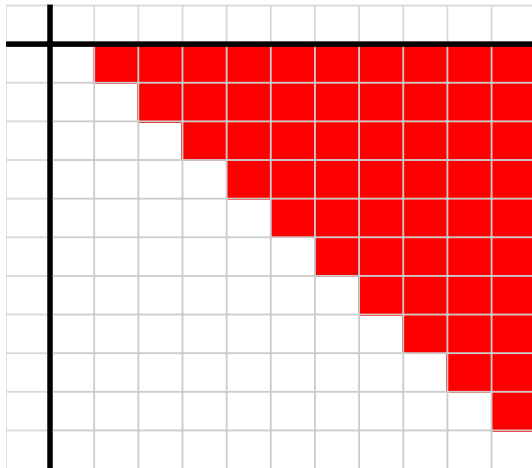
$$\frac{|{"Sw", "we", "ee", "et"} \cap {"Sw", "we", "ea", "at"}|}{|{"Sw", "we", "ee", "et"} \cup {"Sw", "we", "ea", "at"}|} = \frac{|{"Sw", "we"}|}{|{"Sw", "we", "ee", "et", "ea", "at"}|} = \frac{1}{3}$$

Neben diesen Maßen existieren weiter, welche sich an der Aussprache von Worten orientiert. Hierfür ist der Soundex sowie Metaphone ein Beispiel. Beim Soundex wird das erste Zeichen beibehalten und die folgenden Zeichen werden durch die jeweilige Nummer ersetzt. Andere Zeichen werden entfernt. Doppelt aufeinander folgende Zahlen werden auf eine Reduziert. Es werden maximal drei nummer verwendet. Falls weniger vorhanden sind werden sie mit Nullen aufgefüllt.

$$f(n) = \begin{cases} 1 & \text{falls } n \in \{b, f, p, v\} \\ 2 & \text{falls } n \in \{c, g, j, k, q, s, x, z\} \\ 3 & \text{falls } n \in \{d, t\} \\ 4 & \text{falls } n \in \{l\} \\ 5 & \text{falls } n \in \{m, n\} \\ 6 & \text{falls } n \in \{r\} \\ - & \text{sonst} \end{cases}$$

## Vergleichsmatrix

Um jeden String mit jedem anderen zu vergleichen Stellen wir eine sogenannte Vergleichsmatrix auf. Haben wir eine List  $l$  an Strings haben wir eine Matrix  $d$  mit  $|l| \times |l|$  Elementen. Da alle Vergleichsmaße kommutative sind und der Vergleich mit sich Selbs irrelevant ist, müssen wir jedoch nur die Einträge oberhalb der Diagonalen von  $d[0, 0]$  bis  $d[|l|, |l|]$  betrachten. Dies ist in Abbildung 1 dargestellt. Hierbei sind nur die Roten Zellen Interessant und müssen berechnet werden.



**Figure 1: Volle Vergleichsmatrix**

Dabei sieht man auch sofort das Problem der vollen Vergleichsmatrix. Es müssen sehr viele vergliche gemacht werden. Bei 1.000 verschiedener Elemente müssen schon 499.000 bei 10.000 schon 49.990.000 Vergleiche gemacht werden. Dies wird sehr schnell ineffiziente.

Die effizientere Lösung ist hier das Blocking

## SUMMARY

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero

eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

## REFERENCES