Lösungsskizze

Thema: 6 - String Similarity on Flink (Ähnlichkeitsmessung und Performanzanalyse)

Bearbeiter: Christopher Rost
Florian Zeidler

Betreuer: Markus Nentwig

1. Technologien

JDK 8

Java ist eine objektorientierte Programmiersprache, die von Sun Microsystems entwickelt wird. Die Sprache ist derzeit die am häufigsten genutzte Programmiersprache weltweit. Der wesentliche Vorteil von Java ist seine Plattformunabhängigkeit. Da auch die weiteren Technologien (Flink und Simmetrics Framework) auf Javacode basieren, ist auch die Endanwendung plattformunabhängig. Wir benutzen die aktuell stabile Version des JDK 8.

Flink 1.0

Flink ist eine Open-Source Bibliothek um Datenströme in verteilten Systemen zu verarbeiten. Die Engine von Flink bietet dabei Komponenten für die Datenbereitstellung, Transformation, Verteilung der Daten, Aggregierung nach dem Verarbeiten und Fehlerbehandlung. Flink arbeitet dabei mit Datenquellen und -senken, welche eine Vielzahl an Datenformaten annehmen kann. Dadurch ist es möglich, das finale Programm später schnell auf verteilte Cluster zu skalieren. Die aktuelle stable Version ist Flink 1.0.3 (Stand 04.06.2016).

SimMetrics Framework

SimMetrics ist eine Java Bibliothek die verschiedene gängige (Abstands-)Metriken für Zeichenketten implementiert (z.B. Levenshtein Distanz, Cosine Gleichheit, Dice Metrik). Alle Metriken geben dabei einen normalisierten positiven Wert wieder, was sie vergleichbar macht.

2. Problemstellung

Die Datengrundlage für das Projekt besteht aus zwei Geo-Datensätzen, die sich aus Datenquellen von nytimes, dbpedia, geonames und freebase generieren. Die Daten liegen jeweils in drei Dateien im CSV-Format vor.

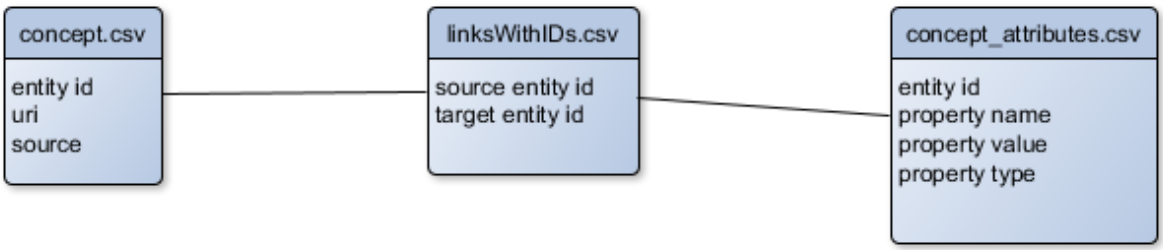

Abbildung 1 Datenschema

Ziel des Programms ist es, alle Datensätze aus der concept_attributes.csv heranzuziehen, die als property name den Wert „label“ aufweisen und deren Zeichenketten des Attributs property value auf Ähnlichkeit bzw. Gleichheit zu untersuchen. Dabei ist der Ansatz der einfachen Stringvergleiche in Java sehr aufwändig und inperformant auf großen Datensätzen. Es sind also geeignete Algorithmen (bzw. Metriken) zu suchen und zu generieren, um auf der Grundlage der Datensätze, möglichst effizient und effektiv, Übereinstimmungen zu finden.

Datenqualität / -struktur

Die beiden Datensätze namens „perfect“ und „linklion“ unterscheiden sich sowohl qualitativ als auch quantitativ. Die in Abbildung 1 dargelegte Struktur gilt für beide Datensätze. Die Zeichenkodierung ist UTF-8.

Der Datensatz perfect weist für die betrachteten Zeichenketten (property value) eine höhere Datenqualität auf, besitzt jedoch weniger Datensätze im Vergleich zu linklion. Mögliche Anpassungen für das Preprocessing wurden für Groß-/Kleinschreibung, verwendete Kommas und zusätzliche Werte in Klammern identifiziert. Vereinzelt sind Kodierungsfehler enthalten.

Im Datensatz linklion treten zusätzlich zu den obigen Beobachtungen die Kodierungsfehler häufiger auf.

3. Vorgehen

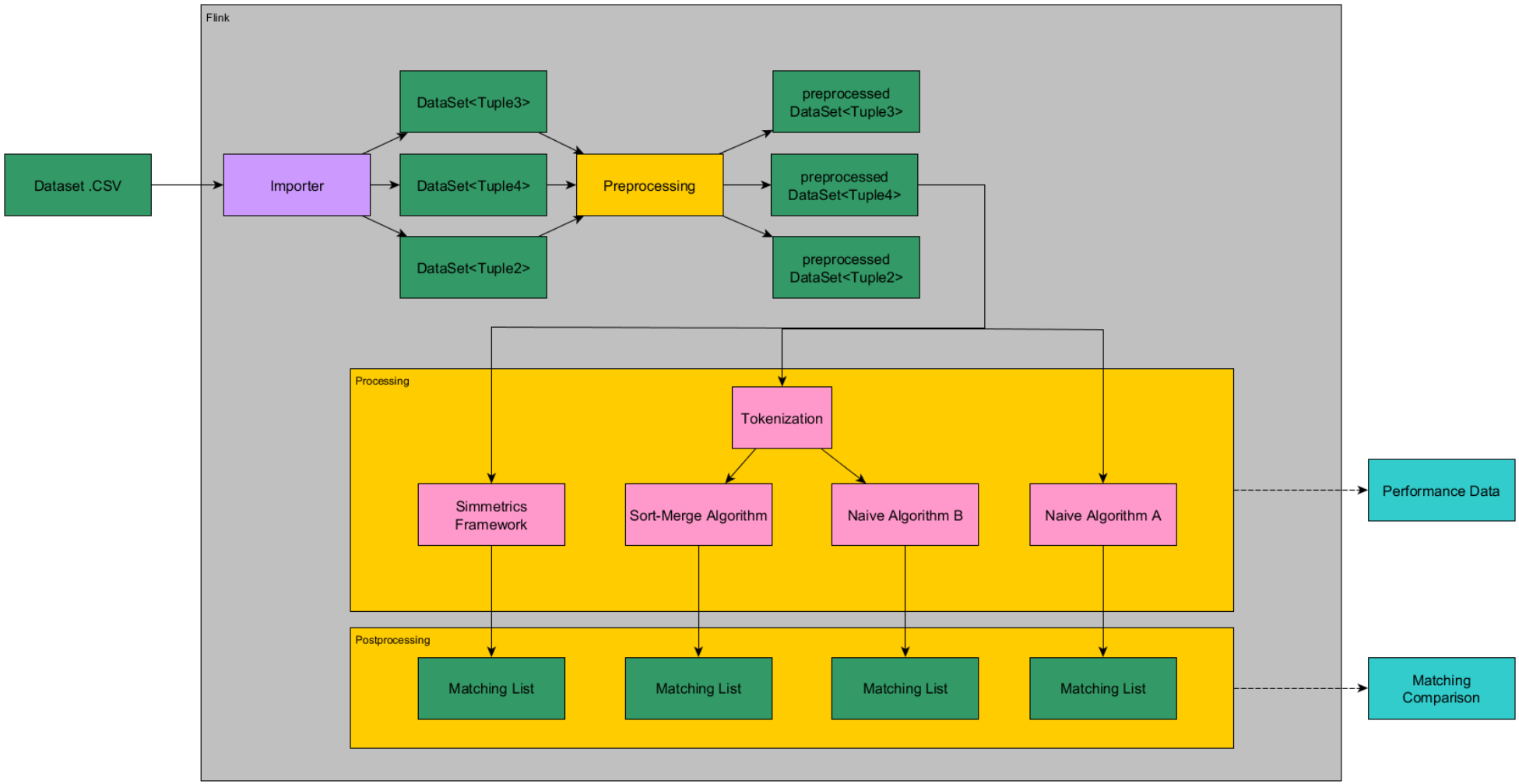

Abbildung 2 Lösungsmodell

3.1 Import

Die im CSV Format vorliegenden Daten werden zunächst durch Datei-basierte Data-Sources in Flink eingelesen und in ein Flink-internes DataSet überführt.

concept.csv -> DataSet<Tuple3<Integer,String,String>>

linksWithIDs.csv -> DataSet<Tuple2<Integer,Integer>>

concept_attributes.csv -> DataSet<Tuple4<Integer,String,String,String>>

Der Importer wird so konzipiert, dass sowohl der linklion, als auch der perfect Datensatz importiert werden kann. Für erste Betrachtungen wird zunächst jedoch ausschließlich der perfect-Datensatz verwendet.

3.2 Transformation (Preprocessing)

Bereits in der Problemstellung wurde auf das Problem der Datenqualität hingewiesen. Um möglichst genaue Treffer zu erzielen werden die importierten Daten bereinigt. Da es sich um einfache Strings handelt, werden alle Zeichen in Kleinbuchstaben umgewandelt, sowie Leerzeichen und Sonderzeichen (Punkte, Komma, Bindestriche) eliminiert. Zusätzlich werden alle Klammern mitsamt deren Inhalt gelöscht.

3.3 Tokenization

Für einige der nachfolgenden Algorithmen ist eine Zerlegung der zu untersuchenden Zeichenkette in Trigramme notwendig. Die Zerlegung wird als Tokenisierung bezeichnet. Dabei wird die Zeichenkette in Fragmente mit 3 Buchstaben zerlegt.

HAUS => __H , _HA , HAU , AUS , US_ , S__ (Unterstriche entsprechen Platzhaltern)

3.4 Algorithmisches Matching (Processing)

Naiver Ansatz A: Beide zu vergleichende Zeichenketten werden naiv mittels String-compare miteinander verglichen. Das Ergebnis (1-true, 0-false) wird zusammen mit den IDs der Zeichenketten gespeichert.

Naiver Ansatz B: Hierbei werden die durch die Tokenisierung erzeugten Triplets beider zu vergleichender Zeichenketten verwendet. Diese werden in Schleifendurchläufen naiv mittels String-compare verglichen. Mit Hilfe der Dice-Metrik wird ein Similarity-Wert berechnet, der zusammen mit den IDs der Zeichenketten gespeichert wird.

„Sort-Merge“: Zunächst werden alle durch die Tokeniserung ermittelten Triplets in ein Wörterbuch überführt. Ein Triplet wird damit ausschließlich durch einen eindeutigen Integer-Wert repräsentiert. Eine zu vergleichende Zeichenkette besteht nun an Stelle einer Menge von Triplets, aus einer identisch großen Menge an Integer-Werten. Die numerischen Werte werden innerhalb dieser Menge aufsteigend sortiert. Anschließend erfolgt die Prüfung auf Gleichheit. Mit Hilfe der Dice-Metrik wird ein Similarity-Wert berechnet, der zusammen mit den IDs der Zeichenketten gespeichert wird.

Simmetrics: Beide zu vergleichende Zeichenketten werden mit Hilfe des Simmetrics-Framework verglichen. Der resultierende Similarity-Wert wird zusammen mit den IDs der Zeichenketten gespeichert.

Bei allen Algorithmen werden die Ausführungszeit, sowie die Speichernutzung für spätere Analysen gemessen.

Erläuterungen zur Dice-Metrik

Um die Gleichheit der Datensätze festzustellen, wird die Dice-Metrik (auch Sørensen–Dice index) herangezogen. Sie setzt die Anzahl der übereinstimmenden Triplets in Relation zur Gesamtzahl der betrachteten Triplets.

$$s = \frac{2 * n_t}{n_x + n_y}$$

n_x ist die Anzahl der Trigramme im String x

n_y ist die Anzahl der Trigramme im String y

n_t ist die Anzahl der übereinstimmenden Trigramme

Damit entsteht eine Metrik $s \in \{z|0 \leq z \leq 1\}$. Wird ein Wert von 0,7 überschritten, werden beide Zeichenketten als gleich angenommen. Zu Analysezwecken kann der Grenzwert in verschiedenen Testdurchläufen variiert werden.

Die Ausgabe der Ergebnisse erfolgt über die Sink-Funktionalität von Flink in eine CSV-Datei je Algorithmus.

3.5 Postprocessing

Bei der Nachverarbeitung werden die erfassten Ergebnisse validiert und inkonsistente Ergebnisse aussortiert. Zusätzlich werden die IDs den entsprechenden Entitäten zugeordnet.

3.6 Auswertung

Die Ergebnisse der einzelnen Algorithmen können nun untereinander, sowie mit den vorgegebenen Datensatz „linksWithIDs.csv“ quantitativ verglichen werden.

Weiterhin wird die Performance der Algorithmen hinsichtlich der Ausführungszeit und Speichernutzung verglichen und analysiert.