Methoden zur Datenanalyse in webbasierten Umgebungen

Michael Aufreiter

Written with Substance
http://substance.io

Zusammenfassung Datenanalyse, also das Erkennen von Mustern bzw. die Suche nach Strukturen in Datenbeständen, ist ein wichtiges Anwendungsgebiet der angewandten Statistik. Dieses Dokument beschreibt den Einsatz von web-basierten Werkzeugen, als Alternative zu etablierten Desktop-basierten Systemen.

1 Einführung

Datenanalyse, oft als Data-Mining bezeichnet bedient sich statistischer Methoden und dient dem Erkennen von Mustern bzw. der Suche nach Strukturen und Besonderheiten in großen Datenbeständen. Datenanalyse wird vor allem in der angewandten Statistik verwendet. Daten werden hierbei in geeigneter Weise zusammengefasst, geordnet und anschließend grafisch dargestellt. Dieser Prozess wird mit einer Vielzahl verfügbarer Werkzeuge erleichtert. Nicht zuletzt wegen des hohen Datenaufkommens handelt es sich hierbei in der Regel um Desktop-basierte Systeme [7].

Die jüngsten Entwicklungen im Bereich der Browser Technologien, wie stark verbesserte Javascript Performance und die Einführung von HTML5 versprechen eine Vielzahl neuer Anwendungsbereiche. Dieses Dokument beschäftigt sich mit der Anwendung von Datenanalyse-Techniken in web-basierten Systemen. Solche Systeme sind nicht länger an lokale Umgebungen gebunden und können, einen geeigneten Browser vorausgesetzt, sofort eingesetzt werden.

2 Fragestellung

Die Einführung web-basierter Systeme zur Datenanalyse wirft eine Reihe von Fragen auf:

– Welche Techniken existieren um diese Aufgabe zu erleichtern?

- Auf welche bestehenden Tools/Frameworks kann zurückgegriffen werden?
- Wie können wiederholte Datenkonvertierungen vermieden werden?
- Wie können Datenoperationen (z.b.Clustering, Filtering, Similarity Measurement etc.) einheitlich implementiert und vielseitig wiederverwendet werden?
- Wie kann man Rechenoperationen auf den Client verlagern um schnelle Antwortzeiten und eine Entlastung der Server Infrastruktur zu erreichen?

3 Zielsetzung

Ziel dieses Dokuments ist einerseits die Erhebung wichtiger Anforderungen, die an ein web-basiertes Framework zur Datenmanipulation gestellt werden, andererseits die Erarbeitung möglicher Lösungen. Diese Lösungsansätze werden im Rahmen eines vom Autor initiierten Praxisprojekts (Data.js) diskutiert und auf deren Praktikabilität überprüft. Bei den beschriebenen Methoden handelt es sich um Lösungsansätze auf unterer Ebene. Konkret wird die Bereitstellung von abstrakten Methoden zur Repräsentation und Manipulation von Datenbeständen auf programmatischer Ebene behandelt. Diese Methoden sollen als Grundlage für die Entwicklung datengetriebener web-basierter Datenanalysewerkzeuge dienen.

4 Anforderungen

Zunächst werden jene Anforderungen spezifiziert, die an ein web-basiertes Datenanalyse-Werkzeug gestellt werden. Zumal es sich hier um ein junges Anwendungsfeld handelt, liegt der Fokus auf Low-Level Funktionalität, die als Basis für Higher-Level Frameworks (wie web-basierten Visualisierungs-Tools) dient. Spence (2007) erarbeite hierfür Kriterien die für diese Aufgabe besonders relevant sind [2]. Aufreiter 2008 nennt weitere Kriterien.

4.1 Datenrepräsentation

Domänen-spezifische Daten weisen unterschiedliche Strukturen auf. Ein Datenanalyse Framework soll geeignete Datenaustausch-formate bereitstellen, um möglichst ohne Umwege Operationen auf den zugrundeliegenden Daten anwenden zu können. Basierend auf den Ausführungen von Thomas und Cook (2005) ergeben sich folgende Anforderungen [1]:

- Beliebige Anzahl an Dimensionen (Eigenschaften eines Datenelements)

- Mehrwertige Eigenschaften
- Ordinale Werte (Kategorische Daten)
- Numerische Werte
- Objekt Werte (Eigenschaft enthält eine Referenz auf weiteres Objekt und legt eine Beziehung fest)

Mit dem Resource Description Framework (RDF) steht ein umfassendes, standardisiertes Datenrepräsentations-Framework zur Verfügung [10]. Leider ist die Werkzeugunterstützung für web-basierte Systeme sehr mangelhaft. RDF erfährt zudem zunehmende Kritik hinsichtlich der hohen Komplexität und der dadurch leidenden Praktikabilität [citation needed].

4.2 Filtern

Filtern ermöglich die gezielte Reduktion des Datenbestands anhand festgelegter Kriterien. Dabei soll der Fokus auf interessante Bereiche gelegt werden, während für diesen Kontext unrelevante Datenelemente ausgeblendet werden [9].

Filtern ist ein wichtiges Instrument zur Datenerkundung (data exploration). Es hilft Benutzern von Datenanalyse-Werkzeugen dabei, interessante Sachverhalte aufzudecken, die aus der Betrachtung der Gesamtdaten nicht ersichtlich sind. Benutzer erlangen so nach und nach Kenntnis über implizite Zusammenhänge. Die Qualität eines Datenanalyse Werkzeugs hängt daher nicht unwesentlich davon ab, welche Möglichkeiten des Filterns angeboten werden.

4.3 Meta-Informationen

.

4.4 Clustering

Clustering wird verwendet um Teilmengen der Datenelemente zu Clustern (Gruppen) zusammenzufassen. Solche Gruppen können sich auf eine oder mehrere ordinale Eigenschafte(n) eines strukturierten Datenbestands beziehen. Mit Clustering geht typischerweise auch das Aggregieren von numerischen Eigenschaften einher. Ein Datenmanipulationsframework sollte daher Methoden bereitstellen, die das Extrahieren von Clustern aus Datenbeständen ermöglichen.

4.5 Effizienz

Effizienter Umgang mit zur Verfügung stehenden Ressourcen (CPU, Speicher) ist ein besonders wichtiges Kriterium für ein Datenanalyse-Werkzeug. Grundopera-

tionen, wie das Auffinden von Datenelementen oder das Ausführen von Mengenoperationen (wie Durchschnitt, Vereinigung, Differenz) sollten besonders rasch durchgeführt werden. Die Repräsentation der Daten im Speicher spielt hier eine wesentliche Rolle. Die Laufzeitumgebung eines Web-Browsers ist event-gesteuert. Langandauernde Berechnungen müssen asynchron erfolgen und die Kontrolle regelmäßig an das User Interface zurückgeben. Sonst drohen Abstürze.

5 Data.js

Data.js ist ein vom Autor initiiertes Datenmanipulations-Framework für Javascript. Es soll das Verarbeiten von Daten im Browser vereinfachen und beschleunigen. Das quelloffene Projekt liegt derzeit in der Version 0.3.0 vor und kann als Basis für datengetriebene Web-Anwendungen (wie z.B. Visualisierungen) verwendet werden. Durch die Beschreibung dieses Projekts sollen mögliche Lösungswege konkret veranschaulicht, diskutiert und evaluiert werden. Dabei sollen die in Kapitel 4 definierten Anforderungen bestmöglich erfüllt werden. Neben einem Interface für Abfragen werden eine Reihe von Operationen bereitgestellt, die für die Datenanalyseaufgaben relevant sind.

5.1 Deklaratives Design

Data.js wurde nach den Prinzipien deklarativen Sprachdesigns entwickelt. Heer und Bostock (2009) schlagen diese Art der Programmierschnittstelle für die Beschreibung von interaktiven Visualisierungen vor. Deklarative Programmierschnittstellen können die Entwicklung vereinfachen und erleichtern das Verständnis durch den Benutzer. Bei dessen Verwendung muss lediglich das gewünschte Ergebnis spezifiziert werden, anstatt festzulegen wie dieses (z.B. durch eine Abfolge von Befehlen) erreicht werden soll. Data.js greift einige dieser Ideen auf, um eine intuitive Programmierschnittstelle zur Abfrage und Manipulation von Daten bereitzustellen.

5.2 Datenformate

Data.js stellt im Wesentlichen zwei Datenaustauschformate bereit: Ein Data.Graph ist ein universelles Containerformat für beliebig komplexe, verlinkte Daten. Eine Data.Collection stellt ein vereinfachtes Interface für tabellarische Daten bereit, während die interne Repräsentation ebenfalls als Graph erfolgt. Es wurde versucht die Komplexität so gering wie möglich zu halten um eine ausdrucksstarke Programmierschnittstelle zu gewährleisten. Die in diesem Abschnitt vorgestellten Formate basieren auf Ideen des Meta Web Object Models [5], werden jedoch in den Kontext client-seitiger Datenverarbeitung gerückt und stellen ein Interface für Javascript bereit.

Data.Graph

```
{
  "/type/person": {
    "type": "type",
    "name": "Person",
    "properties": {
      "name": {"name": "Name", "unique": true, "type": "string", "required: "true"},
      "origin": {"name": "Origin", "unique": true, "type": "/type/location" },
      "age": {"name": "Age", "unique": true, "type": "number"}
    }
  },
  "/type/location": {
    "type": "type",
    "name": "Location",
    "properties": {
      "name": { "name": "Name", "unique": true, "type": "string", "required: "true" },
      "citizens": {"name": "Citizens", "unique": false, "type": "/type/person"}
    }
  },
  "/person/bart": {
    "name": "Bart Simpson",
    "age": "12"
  }
}
```

Die Spezifikation eines Data.Graphs besteht aus Typ- und Objektknoten und im JSON Format notiert. Gemeinsam legen diese das Datenmodell (Schema) sowie deren Instanzen (Daten Objekte) fest. Das beschriebene Format dient als Austauschformat und eignet sich für die Übermittlung von Daten zwischen Server und Client. Sobald die Übertragung abgeschlossen ist wird der Graph mit allen Verbindungen im Speicher aufgebaut. Dies ermöglichlicht schnellen Zugriff und bildet die Basis für komplexe Operationen

Data.Collection

```
var countries_data = {
  "properties": {
    "name": {"name": "Country Name", "type": "string", "unique": true },
    "official_language": {"name": "Official Language", "type": "string", "unique": true },
    "population": { "name": "Population", "type": "number", "unique": true }
  },
  "items": {
    "at": {
```

```
      "name": "Austria",
      "official_language": "German",
      "population": 8356700,
    },
    "de": {
      "name": "Germany",
      "official_language": "German",
      "population": 82062200,
    },
    "us": {
      "name": "United States of America",
      "official_language": "English",
      "population": 310955497,
    }
  }
}
```

;DataTable. [7]. Allerdings werden auch Meta-informationen, zur Beschreibung der Struktur der Daten unterstützt.

5.3 Interne Repräsentation als Graph

Sowohl Data.Graphs als auch Data.Collections werden intern (im Speicher) als Graph repräsentiert. Datentypen mit ihren abstrakten Eigenschaften (Properties) werden jeweils als Knoten im Graph abgelegt und sind entsprechend miteinander verlinkt.

5.4 Verwendung

Ohne auf nähere Details einzugehen wird in diesem Abschnitt die Verwendungsweise von Data.js beschrieben. Dies soll einen grundsätzlichen Einblick in die verwendeten Methoden geben.

Graph Manipulation .

```
// Add Location
graph.set('/location/springfield', {
  name: "Springfield",
  type: '/type/location',
  citizens: ["/person/homer"]
});
```

```
// Add new Person Object
graph.set('/person/homer', {  type: '/type/person',
  name: 'Homer Simpson',
  age: 38,
  origin: '/location/springfield',
});
```

Auf die Daten des Graphen kann jederzeit lesend zugegriffen werden:

```
graph.get('/person/homer').get('age'); // =&gt; 38
```

Abfragen Die Abfrage von Daten erfolgt mit Hilfe von Query-Objekten. Die Objekte (Elemente) eines Datenbestands werden dann mit diesem Query Objekt verglichen. Bei einer Entsprechung wird das Objekt (Datenelement) in das Ergebnis aufgenommen. Um nach allen Bewohnern von Springfield zu suchen wird folgende Abfrage verwendet:

```
var query = {"type": "/type/person", "location": "/location/springfield"};
graph.find(query);
// =&gt; Homer, Marge, Bart
```

]

Clustering Aufgrund der internen Repräsentation als Graph sind Gruppenzugehörigkeiten bereits implizit bekannt. Am Beispiel der Countries Collection lassen sich so beispielsweise alle Objekte (Länder) der Gruppe "Deutsch" ermitteln:

```
var germanSpeakingCountries = countries.properties().get('official_language').values().get('
// =&gt; ["at", "de"]
```

Basierend auf diesem Mechanismus lassen sich beliebige Gruppenzugehörigkeiten (Cluster) erzeugen. Dies geschieht durch die kombinierte Anwendung von Mengenoperationen (Union, Intersect). Üblicherweise ist der Benutzer dabei auch an Aggregaten (z.b. von numerischen Eigenschaften) interessiert. Um beispielsweise die Gesamtpopulationen gruppiert nach Sprache zu erfahren kann die Methode group verwendet werden.

```
var languages = countries.group(["official_language"], {
  "population": {aggregator: Data.Aggregators.SUM, name: "Total Population"}
```

```
});
languages.at(0).get('name'); // =&gt; "German"
languages.at(0).get('population') // =&gt; 90418900
languages.at(1).get('population') // =&gt; 310955497 for "English"
```

Die Gruppierung kann dabei auch über mehrere Eigenschaften erfolgen.

[placeholder]

Type inspection [not yet ready]

```
[not yet ready]
```

Schemainformationen sind genauso wie Datenelemente als Knoten im Graph registriert. Sie können jederzeit gelesen und inspiziert werden.

Verarbeitung großer Datenmengen Bei der Verarbeitung von großen Datenmengen (mehr als 50.000 Datenelemente) stoßen browser-basierte Systeme schnell an ihre Grenzen. Im Rahmen der Entwicklung von Data.js wurden unterschiedliche Methoden evaluiert um auch die Verarbeitung großer Datenbestände zu ermöglichen. Die Wahl fiel dabei auf Streaming (also die asynchrone serielle Verarbeitung von Teilergebnissen). Dies führte nicht zuletzt deswegen zu guten Ergebnissen, da die Kontrolle in regelmäßigen Abständen an den Browser zurückgegeben werden kann. Nach Ablauf einer kurzen Zeitspanne, in der das User Interface aktiv werden kann, wird mit der Verarbeitung des nächsten Teilergebnisses fortgesetzt. Auf diese Weise kann die kurze Idle Phase genutzt werden um den Fortschritt der Operation

5.5 Data Transformers

Data.Transformers dienen als einheitliche Schnittstelle für beliebig komplexe Datenoperationen. Vordefinierte Operationen wie Clustering (group) wurden mit diesem Interface implementiert. Benutzerdefinierte Transformers können jederzeit hinzugefügt werden:

```
Data.Transformers.your_operation: function(graph, [params,...]) {
  // Do data processing
  // return transformedGraph;
}

// Usage
var transformed = Data.Transformers.your_operation(graph, {foo: "bar"});
```

Wichtig ist, dass der ursprüngliche Graph dabei nicht verändert wird. Es werden ausschließlich nicht-destruktive Operationen unterstützt.

6 Evaluierung

Die Evaluierung der Leistungsfähigkeit des vorgestellten Frameworks erfolgt auf Basis einer konkreten Webapplikation, Déjàvis. Es handelt sich dabei um ein Visualisierungstool, das beliebige Datenbestände (sofern als Data.Collection verfügbar) analysieren und mittels Barchart Plots darstellen kann. Für die Darstellung und Manipulation (Filtern, Gruppieren, Aggregieren) der Daten wurde auf Data.js zurückgegriffen.]

6.1 Datenrepräsentation

Getestet wurde mit Daten aus Onlinebestellungen mit mehreren Dimensionen. Nach der Vornahme weniger Optimierungen konnten bis zu 100.000 Datenelemente effizient geladen, gefiltert, gruppiert und als Barcharts gerendert werden. Das Einlesen der Daten (mittels Chunked Streaming) bewegt sich je nach Datenmenge im Bereich 0 bis 20 Sekunden (bei 100.000 Datenelemente).

6.2 Meta-Informationen

Entsprechend der Meta-Informationen einer betrachteten Datenquelle werden in Déjàvis Filter, mögliche Gruppierungseigenschaften (ordinale Werte) sowie numerisch vergleichbare Kennzahlen extrahiert und dem Benutzer in der Benutzerschnittstelle angeboten. Déjàvis hat die Visualisierung beliebig strukturierter, tabellarischer Daten zum Ziel. Der Zugriff auf Meta-Informationen ist aus diesem Grund unverzichtbar. Data.js konnte diese Anforderungen erfüllen und ermöglicht der Anwendung den Zugriff auf alle relevanten strukturellen Daten.

6.3 Filtern

Déjàvis verwendet das Prinzip von Faceted Search zum Filtern und Erkunden von Daten. Basierend auf den zugrundeliegenden ordinalen Werten eines Datenbestands werden Facetten extrahiert, nach der Häufigkeit ihrer Vorkommen sortiert, und dem Benutzer als Filter angeboten. Nach jeder Filter-Operation werden die Facetten (basierend auf dem reduzierten Datenbestand) neu berechnet um den Benutzer intuitiv durch den Datenbestand führen zu können. Die Implementierung basiert auf der Verwendung von Mengenoperationen (Union,

Intersect) für die Berechnung der Facetten und der Anwendung von Abfragen, die sich aus den aktuell festgelegten Filtern ergeben.

Filteroperationen bewegen sich im Zehntelsekunden bis Sekundenbereich und sind somit ausreichend schnell um eine flüssige Bedienung zu gewährleisten.

6.4 Clustering

;group verwendet. Auch hier konnten äußerst zufriedenstellende Ergebnisse erzielt werden. Das Gruppieren von 50.000 Datensätzen nach ein oder mehreren Eigenschaften dauert etwa eine halbe Sekunde. Die subjektiv empfundene Antwortzeit lässt sich mit der einer vergleichbaren Operation in Microsoft Excel vergleichen.

7 Conclusio

Die Referenzimplementierung konnte den definierten Anforderungen weitestgehend genügen. Nach intensiven Tests mit Echtdaten stellte sich der web-basierte Ansatz für kleine bis mittelgroße Datenmengen als praktikabel heraus. Es konnte gezeigt werden, dass der Graph-basierende Implementierung, zusammen mit leichtgewichtigen Grundoperationen zu zufriedenstellenden Ergebnissen führt.

Die Limitierungen der browser-basierten Laufzeitumgebung schränken den Spielraum jedoch ein. Zwar ist parallele Datenverarbeitung in mehreren Threads möglich (Webworker API), aus Sicherheitsgründen kann jedoch kein Shared Memory Bereich genutzt werden. Die Kommunikation zwischen mit den Worker Threads ist stark limitiert, daher eignen sich diese nur für bestimmte Aufgaben.

Generell gilt jedoch, dass Einschränkungen weniger die Laufzeitumgebung betreffen, als das Fehlen von geeigneten Softwarekomponenten, die auf anderen Plattformen längst zur Verfügung stehen. Die Entwicklung von Data.js soll dazu beitragen diesem Umstand entgegenzuwirken und geeignete Komponenten zur Entwicklung von performanten Datenanalysewerkzeugen bereitstellen. Javascript erlebt derzeit eine Renaissance, und kann seit einiger Zeit auch serverseitig eingesetzt werden (Node.js, Rhino). Dieser Umstand und eine stark wachsende Community an Javascript Entwicklern lassen für diese Plattform in Zukunft einiges erhoffen.

8 Literatur

[1] J. Thomas, K. Cook — Illuminating The Path: The Research and Development Agenda for Visual Analytics

[2] Robert Spence 2007 — Information Visualization

[3] Google Visualization Toolkit

[4] Heer and Bostock 2009 —Declarative Language Design for Interactive Visualization

[5] MQL Reference — Metaweb Architecture

[6] Michael Aufreiter 2008 — Informationsvisualisierung und Navigation im Semantic Web

[7] The R Project for Statistical Computation

[8] RDF — Resource Description Framework

[9] Freitas, C. M. D. S., P. R. G. Luzzardi, R. A. Cava, M. A. A.Winckler, M. S. Pimenta und L. P. Nedel: Evaluating Usabilityof Information Visualization Techniques