BDLC Gruppenarbeit

Datenquelle: SBB Zugverkehrsinformationen

Quelle: https://commons.wikimedia.org/wiki/File:BigData_2267x1146_white.png

Hochschule Luzern - Informatik
Modul BDLC – Big Data Lab Cluster
FS19

Gruppenmitglieder
Becker Maximilian
Csetreki Sándor
Häfliger Andrea
Schär Tobias

Rotkreuz, 14. Mai 2019

Inhaltsverzeichnis

1 Kurbeschrieb dieser Dokumentation

Als Teil unserer Prüfungszulassungsaufgaben haben wir von unserem Dozententeam den Auftrag erhalten, selbstständig ein Big Data-Problem zu identifizieren, dieses zu analysieren und mit den vermittelten Unterrichtinhalten bestmöglich zu bearbeiten.

Unser Team besteht aus Andrea Häfliger, Max Becker, Sándor Csetreki und Tobias Schär – alle haben einen kaufmännischen Abschluss oder eine Lehre als Mediamatiker abgeschlossen. Keiner von uns hatte vor dem Modulbesuch BDLC je etwas selbst mit Big Data zu tun, geschweige denn selbst irgendwelche Versuche getätigt, das Thema selbstständig zu erlernen. Somit tauchten wir dann im Februar dieses Jahres zwar mit viel Begeisterung aber auch mit ein wenig Ungewissheit in dieses fremde Themengebiet ein.

Auf den folgenden Seiten wollen wir Euch aufzeigen, was wir in den letzten Wochen und Monaten erleben durften und wie wir die gestellte Aufgabe im Rahmen unserer Teamarbeit abgewickelt haben. Schon mal so viel voraus: Wir durften sehr viel dazulernen!

Kleiner Hinweis für Interessierte: Ziel unseres Projektes war es auch, die ganzen Konfigurationen zum einfachen Nachspielen in einer VM analog zum Cluster hcpelb direkt in unser deployment-Script einzubauen, damit eine Installation in zwei einfachen Schritten und ohne grosse Hindernisse möglich ist. Hierzu ist Folgendes zu tun:

1. ZIP-Ordner entzippen.
2. Install.sh ausführen (geht direkt auf der VM – Verbindung mit Cluster wird selbst hergestellt).
3. Falls Zeppelin-Notebook nicht gehen sollte, Cluster nochmals neu starten. Im Rahmen des deployment-Scripts wurden einige Module auf dem Cluster installiert.

Nun wünschen wir auf jeden Fall viel Vergnügen beim Durchstöbern unserer erarbeiteten Inhalte!

2 Datenquelle

Vorweg zur eigentlichen Gruppenarbeit und auch zu Beginn des Moduls wurden uns durch Herrn Grossniklaus diverse mögliche Anwendungsbereiche zu Big Data aufgezeigt und erklärt, wann man überhaupt von «Big Data» sprechen darf und wann nicht. Grundsätzlich kann man vereinfacht sagen, dass «Big Data» dort entsteht, wo Daten aus vielen Schnittstellen zusammenfliessen sollen und der Aufwand einer zentralen Datenlagerung nicht realistisch oder auch nicht wirtschaftlich ist. Wir haben uns also schnell darauf geeinigt, dass wir den Fokus auf irgendeinen Bereich legen könnten, bei dem ebendiese Problematik vorliegt – und somit konnten wir dann auch relativ schnell eine entsprechende Datenquelle identifizieren.

Die SBB stellt seit wenigen Jahren eine kostenfreie API zur Verfügung, welche verschiedenste Metriken rund um Züge, Bahnhöfe, Verzögerungen und noch viele weitere Dinge bereitstellt – hier sprechen wir von gegen 100 verschiedenen Datenquellen, die Interessierten frei zugänglich und unbeschränkt zur Verfügung stehen. Wir haben uns nach kurzer Diskussion und genauerer Betrachtung der einzelnen Datenstämme auf der entsprechenden Plattform (https://data.sbb.ch/pages/home/) dazu entschieden, Zugverkehrsinformationen (genauer gesagt alle Störungen des Schweizer Bahnverkehrs) zu untersuchen. Diese Datenquelle wählten wir aus, da wir viel Potential für die Aufgabenstellung sahen und uns dies als Pendler auch direkt betreffen. Auch wir sind immer wieder einmal von Ausfällen betroffen und hoffen durch die Bearbeitung etwas Interessantes herauszufinden. Die Datenquelle wird täglich aktualisiert und liefert JSON-Dateien.

Mögliche Fragestellungen

Wir hatten somit eine Datenquelle, doch welche Informationen wollen wir aus dieser gewinnen? Da wir im Rahmen dieses Moduls keine Raketenwissenschaft betreiben müssen, haben wir uns auf augenscheinlich kleinere Fragen beschränkt, dennoch aber auch daran gedacht, dass wir mehr interpretieren wollen, als dies ein einfaches Excel könnte.

Initial haben wir somit folgende Fragestellungen erarbeitet:

- Wie viele Störungen gibt es pro Woche?
- An welchen Wochentagen treten die meisten Störungen auf?
- Wo gibt es regelmässig Störungen?
- Welche Störung tritt am meisten auf?
- Wie lange dauert es, bis Störungen behoben sind

3 Datenfluss, Systemarchitektur & verwendete Methoden

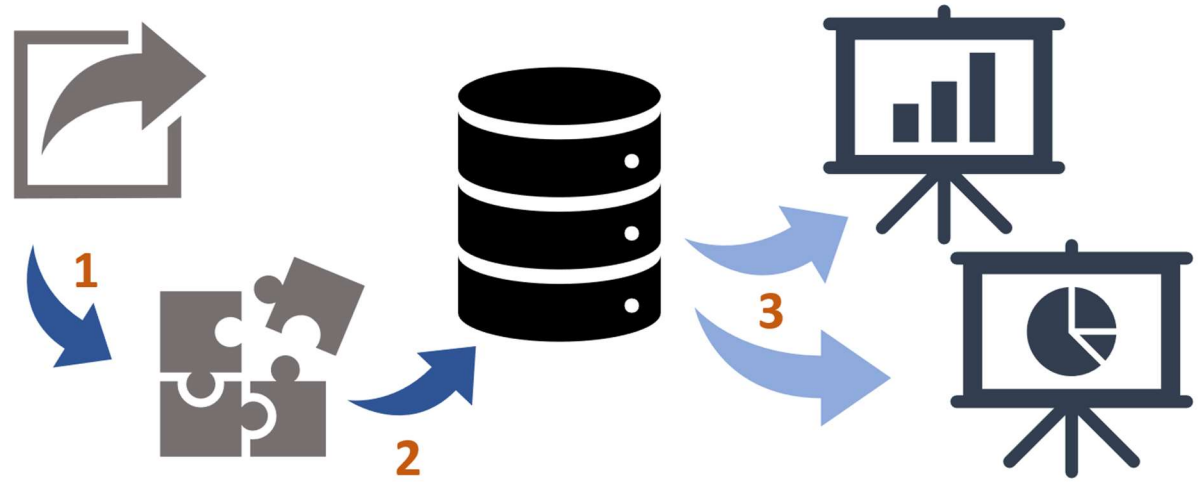


Abbildung 1: Vereinfachter Ablauf unseres Big Data-Problems

Wir haben uns zu Beginn des Projektes Zeit genommen, eine generelle Vorgehensweise im Team festzulegen. Die Produkte dieser Konzeptionsarbeit kamen uns in den nachfolgenden Schritten sehr gut entgegen. Auch wenn wir anfangs nicht genau wussten, welche Technologie wir wann einsetzen wollen, haben wir den obenstehenden Grundablauf beachten wollen und haben diesen auch für eine einfache Gliederung der Arbeitspakete gebrauchen können.

Wir fassen kurz zusammen: Die Datenquelle wird täglich aktualisiert, die Datenmenge ist überschaubar und die Komplexität der Fragestellungen hält sich in Grenzen. Dies schreit förmlich danach, dass wir auch ein entsprechend einfaches Setup für unseren Feldversuch verwenden können. Folgend haben wir in einigen Sätzen die grundsätzliche Vorgehensweise aufgeführt. – detaillierte Vorgehensweisen können problemlos dem Source Code entnommen werden.

3.1 Herunterladen der Daten über entsprechende API

Mittels eines Bash-Skripts ziehen wir die benötigten Daten von der bereitgestellten Datenschnittstelle herunter. Hierbei mussten wir eigentlich nur schauen, dass nur Informationen gezogen werden, welche wir auch verwenden möchten. Dies haben wir mit einem cronjob soweit automatisiert, dass die entsprechenden Dateien einmal pro Tag als JSON-Datei heruntergeladen und verarbeitet werden kann.

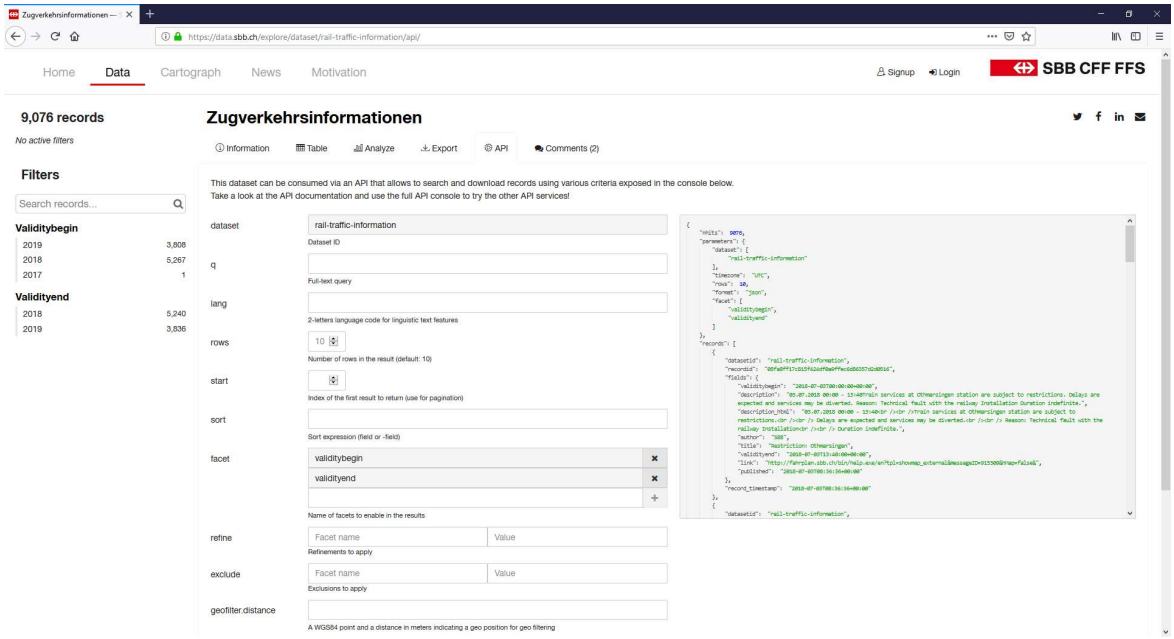
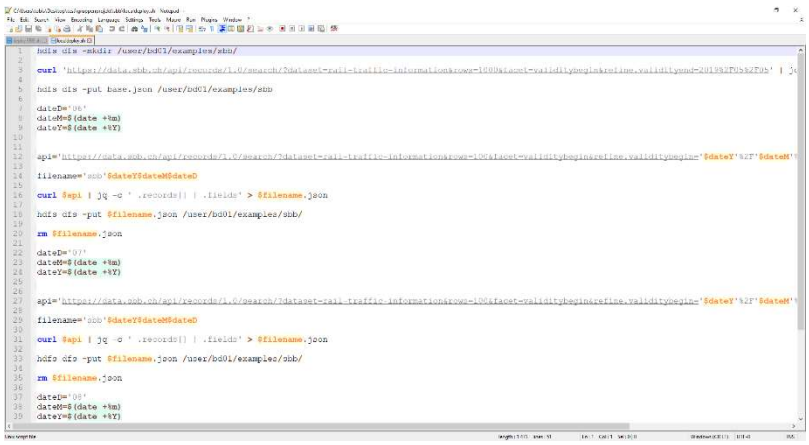


Abbildung 2- Die SBB-API im Überblick

Grobkörnige Aufbereitung der Daten

Bei diesem Schritt war relativ schnell klar, dass wir Bash verwenden würden. Die Curl-Commands sind einfach zu verwenden und ermöglichen es auf einfache Weise Daten grob zu sortieren – genau dies haben wir so auch umgesetzt, um beispielsweise unnötige Metadateien in unseren Files zu haben. Der Inhalt wird direkt mit jq geparst, sodass nur die *fields* Objekte und deren Werte übrigbleiben. Diese Daten haben wir dann in einem eigenen JSON-File abgespeichert.



```
hdfs dfs -mkdir /user/bd01/examples/sbb/

curl 'https://data.sbb.ch/api/records/1.0/search/?dataset=rail-traffic-information&rows=1000&facet=validitybegin&refine.validityend=2019%2FU%5B%2FU%5' | j

hdfs dfs -put base.json /user/bd01/examples/sbb

dateD='06'
dateM=$(date +%m)
dateY=$(date +%Y)

api='https://data.sbb.ch/api/records/1.0/search/?dataset=rail-traffic-information&rows=100&facet=validitybegin&refine.validitybegin='$dateY'%2F'$dateM''

filename='sbb'$dateY$dateM$dateD
curl $api | jq -c ' .records[] | .fields' > $filename.json

hdfs dfs -put $filename.json /user/bd01/examples/sbb/

rm $filename.json

dateD='07'
dateM=$(date +%m)
dateY=$(date +%Y)

api='https://data.sbb.ch/api/records/1.0/search/?dataset=rail-traffic-information&rows=100&facet=validitybegin&refine.validitybegin='$dateY'%2F'$dateM''

filename='sbb'$dateY$dateM$dateD

curl $api | jq -c ' .records[] | .fields' > $filename.json

hdfs dfs -put $filename.json /user/bd01/examples/sbb/

rm $filename.json

dateD='08'
dateM=$(date +%m)
dateY=$(date +%Y)
```

Abbildung 3 - Unser localDeploy.sh

3.2 Abspeichern auf dem Cluster

Die grob strukturierten JSON-Files müssten nun irgendwo abgelegt werden – und hier haben wir bereits eine erste Grundentscheidung zu fällen: Wollen wir die entsprechenden Datensätze in Hive-Tabellen abspeichern oder ist die aktuelle Datenmenge auch gut «händisch» zu beherrschen? Wir haben uns hier im Verlauf der beiden Wochen für die zweite Möglichkeit entschieden.

Datenablage auf HDFS

Die Speicherung der Daten in Hive macht in unserem Fall keinen Sinn, da dies lediglich ein Umweg ist, welcher zusätzlich unnötig Speicher frisst. Die bezogenen Daten werden aus diesem Grund direkt im Hadoop Distributed File System (HDFS) abgelegt. Es ermöglicht uns einen schnellen Zugriff auf Daten und wir wissen auch mit wenig Aufwand, wo wir die Daten zu löschen haben, falls wir sie nicht mehr benötigen sollten. Hive würde im Kontext von Big Data aber definitiv schnell ein Thema werden, sobald grössere Datenmengen effizient abgespeichert und verteilt abgerufen werden sollten – für unser Experiment reicht jedoch das direkte Abspeichern auf dem Gateway locker aus.

3.3 Analyse

Unser JSON-File bietet diverse interessante Informationen, welche es nun zu analysieren gilt. Die entsprechenden Fragestellungen werden unter dem Kapitel 1 dieser Dokumentation genauer erläutert. Die Ergebnisse dieser Analyse werden separat geloggt und können somit dann auch für weitere Recherchen wie Trendanalysen verwendet werden.

Panda
Das Python-Framework Pandas liefert eine grossartige Möglichkeit effizient und einfach Dataframes mit Python-Code anzulegen. Dataframes können des Weiteren sehr einfach mit weiteren Pandas-Funktionen beliebig aufbereitet werden und somit auch als perfektes Reporting-Tool genutzt werden kann. Dies haben wir im Rahmen unseres Projekts auch so angewandt.

Zeppelin
Als Aggregations- und Präsentationsplattform nutzen wir Zeppelin. Nicht nur können wir hier einzelne Code-Zeilen ausführen, sondern zeitgleich auch ohne grosse Aufwände einfache Reportings mit SQL und den integrierten visuellen Anzeigemöglichkeiten einbetten. Die Daten in Zeppelin werden zuerst von einem JSON-File in ein Pandas-Dataframe geladen und danach mit SQL-Befehlen in Zeppelin so angepasst, wie wir diese gerade verwenden können.

4 Prototyp

Die erste detaillierte Sichtung der Datensätze aus der SBB-API hat ergeben, dass diese nicht immer sauber genug sind, um direkte Analysen durchzuführen. Wir haben schnell bemerkt, dass Informationen, wie die Störungsursachen, geschätzte Störungszeiträume und auch Streckeninformationen im Feld «description_hmtl» versteckt sind und dies nicht immer in derselben Reihenfolge. Wir mussten somit ziemlich schnell mit mittels «Regular Expressions» bewerkstelligt diese Informationen möglichst sauber aus den Datensätzen extrahieren.

Im der folgenden Abbildung ist ersichtlich, dass die Ursache der Störung nicht immer an derselben Stelle im Feld auftaucht: Hier einmal nach dem sechsten *break* und einmal nach dem zehnten. Beim genaueren Hinsehen konnten wir ein Muster beim Verwenden von Breaks in den html-Formatierungen im Feld «description-html» erkennen, welches (abhängig vom Störungstyp) eine bestimmte Reihenfolge der Feldinhalte bestimmte.

```
1.  "description_html": "21.06.2018 00:00 - 20:00<br />
2.  <br />
3.  Between Genève and Nyon only limited train services are operating.<br />
4.  <br />
5.   Delays and cancellations are expected.<br />
6.  <br />
7.   Reason: Technical fault with the railway Installation<br />
    <br />
    Duration  indefinite.",



1.  "description_html": "20.06.2018 00:00 - 22:30<br />
2.  <br />
3.  Between Bern and Olten only limited train services are operating.<br />
4.  <br />
5.   The trains IC 1, 6, 8, 21, 61 are being rerouted.<br />
6.   The trains IR 15, 16, 17, 26 are being rerouted.<br />
7.  <br />
8.   The trains IC 21 + IR 26 make an unscheduled stop in Liestal.<br />
9.  <br />
10.  Please allow for a longer travelling time.<br />
11. <br />
12.  Reason: Track blocked by train.<br />
    <br />
    Duration  indefinite.",
```

Abbildung 4: Inkonsistente Beschreibungen

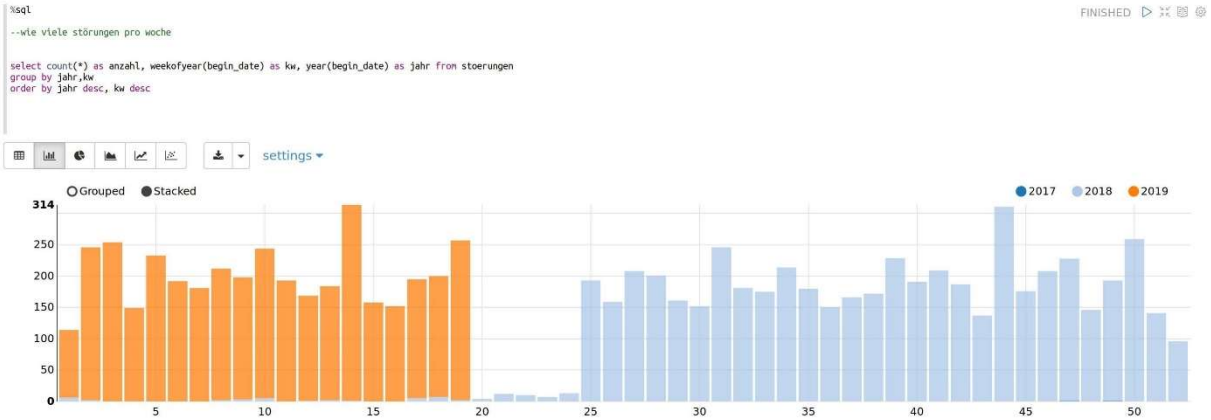
Für die Aufbereitung der Daten mussten wir entsprechende Expressions zusammenstellen, welche die Informationen brauchbar extrahieren – so konnte auch beispielweise die Störungsart aus den meisten Datensätzen herausgelesen werden.

5 Ergebnisse

Nachdem die Daten alle geparst und in eine View gepackt wurden, konnten wir uns unseren Fragestellungen widmen. Die jeweiligen Daten wurden in Zeppelin mittels PySpark über Befehle in der SQL-Sprache extrahiert. Alle entsprechenden Diagramme können mit den SQL-Queries im Zeppelin-Notebook nachgespielt werden.

Fragestellung 1:
Wie viele Störungen gibt es pro Kalenderwoche?

In dieser Grafik ist schön zu erkennen, ab wann die SBB-API richtig Daten erfasst hat, denn für die Kalenderwochen vor KW25 im Jahr 2018 bestehen nur sehr wenige Messungen.



Zwar konnten wir hier nicht einen klaren Trend erkennen, aber waren trotzdem überrascht, dass in in den Neujahrswochen tendenziell weniger los ist.

Fragestellung 2:
An welchem Wochentagen treten die meisten Störungen auf?

Für diese Fragestellung haben wir alle Meldungen nach Wochentag geordnet und dann gezählt. Folgende Darstellung kam dabei heraus:

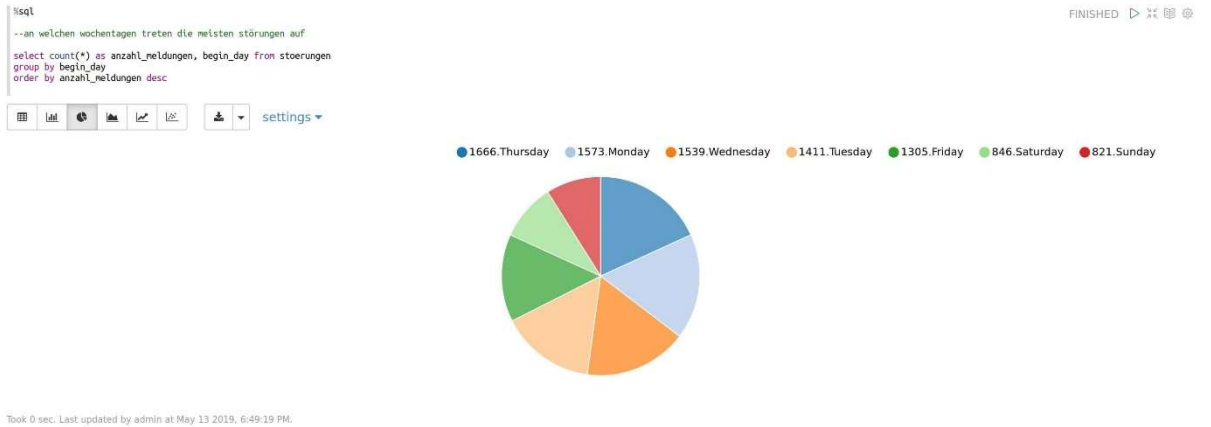


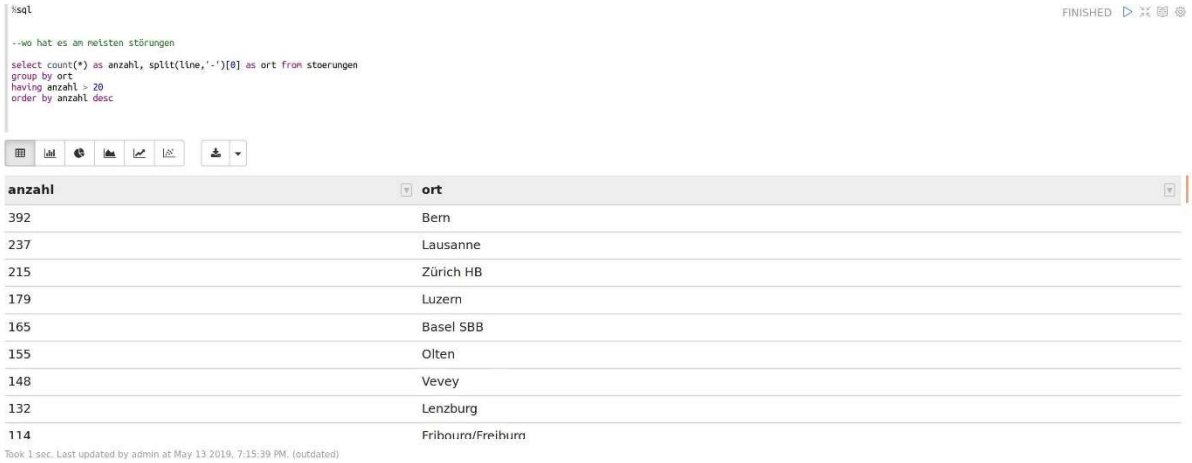
Abbildung 5: Anzahl Meldungen nach Wochentagen

Am Wochenende gibt es jeweils halb so viele Meldungen wie zu den Spitzenzeiten Montag und Donnerstag. Unter der Woche gibt es aber täglich mindestens 1.5 – 2-mal so viele Störungen.

Alles in allem gibt es hier keine grossen Überraschungen. Bemerkenswert ist vielleicht, dass entgegen der Intuition nicht am Montag, sondern am Donnerstag die meisten Störungen aufzutreten scheinen.

Fragestellung 3:
Wo gibt es regelmässig Störungen?

Hier haben wir analysiert, welche Stationen in den Störungsmeldungen enthalten sind, also nicht die ganzen Strecken zwischen den Endbahnhöfen, sondern effektiv genannte Bahnhöfe.



```
%sql

--wo hat es am meisten störungen

select count(*) as anzahl, split(line,'-')[0] as ort from stoerungen
group by ort
having anzahl > 20
order by anzahl desc
```

FINISHED

anzahl	ort
392	Bern
237	Lausanne
215	Zürich HB
179	Luzern
165	Basel SBB
155	Olten
148	Vevey
132	Lenzburg
114	Fribourg/Freiburg

Took 1 sec. Last updated by admin at May 13 2019, 7:15:39 PM. (outdated)

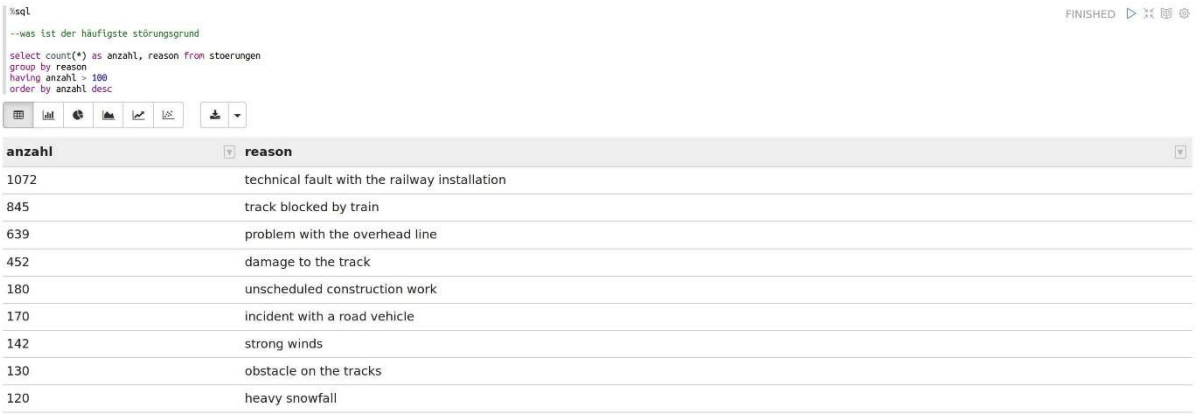
Abbildung 6 - Störungen nach Ort

Mit Abstand am meisten Vertreten ist Bern. Dies könnte man auf die Beanspruchung der Strecke zurückführen, wenn man bedenkt, dass Bern und auch Olten wichtige Kreuzpunkte für Züge aus allen möglichen Richtungen (u.a. Zürich, Basel, Biel und Luzern) darstellen.

Ebenfalls oft vertreten ist Lausanne, ein Bahnhof, der von jeder Linie passiert wird, auf der ein Zug nach Genf oder von Genf wegfährt. (Ausnahme bilden einige Linien von Neuchâtel und Yverdon Richtung Genf). Weiter sind Luzern und Vevey auffällig oft vertreten – Zürich (nicht überraschend) ebenfalls als grosser Bahnhof.

Fragestellung 4:
Welche Störung tritt am meisten auf?

Die Top 4 der meist gemeldeten Gründe für Störungen deckt zwei Drittel aller Meldungen ab. Diese betreffen die Infrastruktur der SBB selber: Gleise, Züge und deren Stromversorgung sind das Problem. Die meisten aller Probleme treten also ohne Einwirkung der Umwelt auf.



```
%sql
--was ist der häufigste störungsgrund

select count(*) as anzahl, reason from stoerungen
group by reason
having anzahl > 100
order by anzahl desc
```

FINISHED

anzahl	reason
1072	technical fault with the railway installation
845	track blocked by train
639	problem with the overhead line
452	damage to the track
180	unscheduled construction work
170	incident with a road vehicle
142	strong winds
130	obstacle on the tracks
120	heavy snowfall

Abbildung 7 - Störungsarten

Alle anderen Ergebnisse geben einen Interessanten Einblick über die restlichen Probleme, mit der die SBB zu kämpfen hat. So sind Windböen beispielsweise ein grösseres Problem als Gegenstände, die auf der Strecke liegen und Autounfälle die häufigste Ursache für Störungen äusserer Einwirkung.

Fragestellung 5:
Wie lange dauert es, bis Störungen behoben sind?

Im folgenden Plot-Chart soll aufgezeigt werden, dass mehrtägige Störungen doch nicht so selten erscheinen, wie wir vielleicht denken mögen.

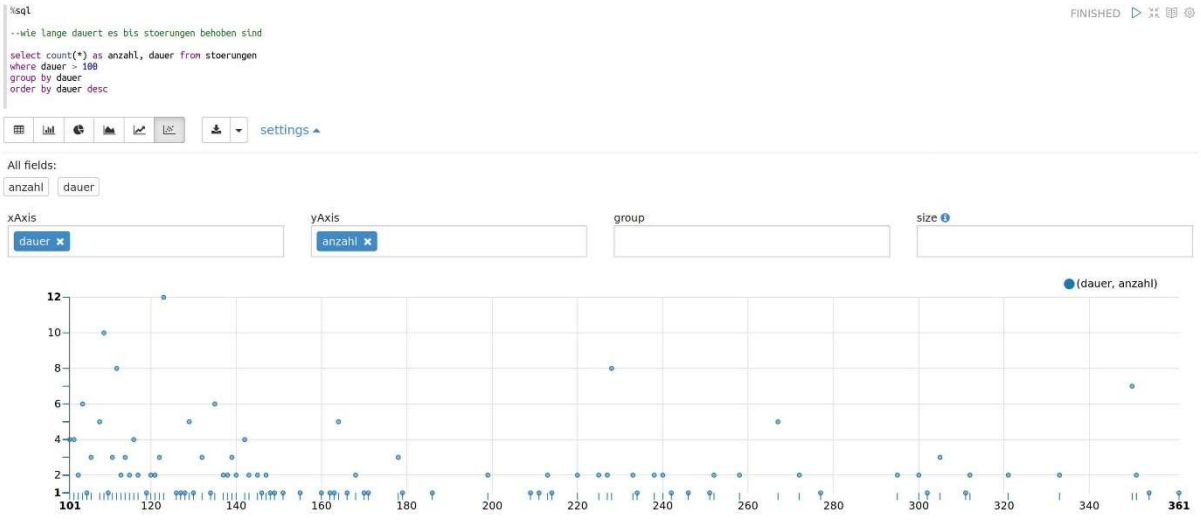


Abbildung 8 - Dauer von Störungen

Fairerweise muss man auch sagen, dass es sich bei diesen Unterbrüchen (nach stichpunkthafter Untersuchung durch uns) um Bauarbeiten handelt und nicht um effektive Un- oder Ausfälle.

6 Gedanken zu den Ergebnissen

Die erarbeiteten Resultate zeigten uns auf, dass bei einem Big Data-Problem nicht zuerst eine Fragestellung gestellt wird und danach gezielt diese Daten gesucht werden, sondern in der Praxis oft andersrum gedacht wird. So haben auch in unserem Beispiel die Daten der SBB-API nicht perfekt auf unsere Fragestellung gepasst, sondern wurden einfach mal irgendwie gespeichert. Dies wird wohl auch in der Praxis oft so sein: Daten werden gesammelt und erst aus diesen Daten sollten wir sinnvolle Ergebnisse ziehen.

Wir haben uns im Zusammenhang mit unseren Resultaten auch gefragt, ob die Vorgehensweise der Datenflüsse die Richtige wäre, auch wenn die Datenmenge stark skalieren würde. Hierzu konnten wir keine klare Antwort finden. Logisch ist, dass ab einer gewissen Grösse die Verarbeitung im HDFS mit verschiedenen Worker-Nodes (analog zum hcpelb-Cluster) besser und schneller verarbeitet werden könnte – speziell dann, wenn verschiedene Quellen in verschiedenen Files zusammengeführt werden sollen und dedizierte Echtzeitreports erstellt werden müssten. Die Frage bleibt aber, ob die Daten der vorliegenden API überhaupt ein Cluster benötigen.

7 Gruppenorganisation

Da niemand aus unserer Gruppe das Vorgänger-Modul BDL besuchte und auch sonst keine Kenntnisse zu diesem Thema vorhanden waren, erarbeiteten wir vieles gemeinsam und konnten die Arbeitspakete nur bedingt aufteilen. Zur Abwicklung der einzelnen Aufgaben haben wir uns stets in der Schule getroffen, um direkte Kommunikation zu ermöglichen und neue Kenntnisse direkte mit den anderen zu teilen. Treffen haben wir über unserer Whatsapp-Gruppe vereinbart – die entsprechenden Termine können dem angehängten Arbeitslogbuch entnommen werden.

Wir haben zur einfachen Verwaltung unserer Codes zusätzlich ein GitLab-Repo verwendet, damit wir geänderte Files einfach miteinander austauschen konnten. Für textbasierende Dokumente haben wir einen geteilten OneDrive-Ordner verwendet.

Die identifizierten Arbeitspakete und entsprechende Aufwände können dem separaten Excel-Sheet entnommen werden.

8 Erfahrungen

Zu Beginn unserer Gruppenarbeit haben wir uns einige Minuten Zeit genommen, um mögliche Risiken der bevorstehenden Arbeiten ehrlich und konkret zu identifizieren:

- Was, wenn sich unser Problem nicht als ein «Big Data»-Problem entpuppt?
- Was, wenn wir nicht wissen, wie wir unser Problem angehen müssen?
- Was, wenn wir den Auftrag aus diesem Grund nicht zeitgerecht erledigen können?

Die erste Frage beschäftigte uns nach dem Verstreichen der Hälfte der Zeit noch immer, also haben wir uns bei Herrn Grossniklaus schlau gemacht. Wir setzen danach die Gruppenarbeit so auf, dass klar ersichtlich sein soll, dass es sich hier nicht um eine fertige Lösung handelt, sondern mehr um

einen möglichen Lösungsansatz für ein Gesamtsystem, welches dann aber um einiges verstrickter wäre.

Umgang mit potenziellen Risiken
Die grösste Herausforderung bestand darin, den Wissensstand der einzelnen Gruppenmitglieder bestmöglich zu nutzen. Man merkte vor Allem zu Beginn der Gruppenarbeit, dass wir alle auch über komplett verschiedene Kenntnisstände in Big Data verfügten. Wir haben uns gerade auch deshalb dazu entschlossen, stets alle Arbeitsschritte gemeinsam in der Gruppe anzugehen – auch wenn dies im Gegenzug einen massiv grösseren Zeitaufwand bedeutete. Bei gewissen Entscheidungen hat es sich definitiv gelohnt, dass vier Köpfe gleichzeitig ein bestimmtes Problem von vier verschiedenen Seiten aufgriffen.

Die Projektdauer von zwei Wochen wurde von uns schon zu Beginn der Gruppenarbeit als knapp empfunden – wir sind froh, dass wir nun aber trotzdem etwas Brauchbares vorzeigen können.

Unser Gruppenfazit
Rückblickend konnte durch diese Vorgehensweise aber sicherlich jedes Teammitglied weitere Erfahrungen gewinnen. Es war für uns definitiv auch spannend zu sehen, dass jeder seine eigenen Erfahrungen gewinnbringend in das Projekt einbringen konnte und schlussendlich ein sauberes, wenn auch überschaubares Projektlein entstand.

Was uns besonders geholfen hat war das mutige Vorgehen, stets möglichst alle Schritte gemeinsam zu absolvieren, um maximal von sämtlichen Tätigkeiten profitieren zu können. Den damit verbundenen Mehraufwand und das somit auch sportliche Zeitmanagement konnten wir einigermassen gut im Zaun halten, allerdings haben wir den Druck beim Fertigstellen jederzeit gut bemerkt (v.A. in der KW19). Und diese Erfahrung (auch wenn sie evtl. einen kleinen Nachgeschmack auf die Beurteilung unserer Arbeit haben wird) werden wir mitnehmen. Wir sind uns sicher, dass diese Vorgehensweise für unsere Teamkonstellation genau die Richtige war.

Was würden wir das nächste Mal anders angehen?
Das erste Mal eine Big Data-Problemstellung selbst zu bearbeiten war für niemanden von uns einfach. Wir hatten mit diversen kleineren Stolpersteinen und etlichen Stunden auf Internetforen aber definitiv einen Erfahrungsschatz angesammelt, welchen wir bei ähnlichen künftigen Projekten gleich verwenden können.

Die generelle Vorgehensweise würden wir beibehalten – einzig wäre die Möglichkeit zu erwägen, ob man erst die Datengrundlage besser analysieren möchte und erst danach potenzielle Fragestellungen daraus abstrahieren möchte. Jeder von uns nimmt aber sicherlich auch seine eigenen Learnings mit und die Kenntnis, dass Big Data eigentlich eine ziemlich grossartige Sache ist!

9 Abbildungsverzeichnis