1. Testat

Konzeptioneller Entwurf einer Anwendung zum Thema:

Tweet Analyse von News

Bearbeitet von:

Hammann, Felix

Hartwig, Mattis

Mäder, Hannes

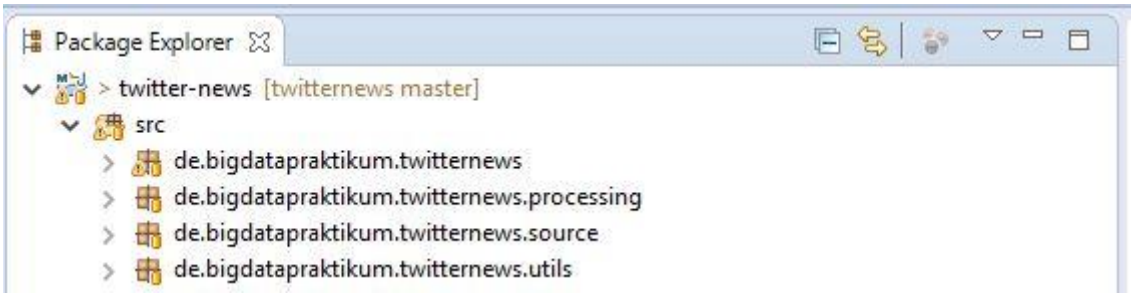
Betreuender Hochschulmitarbeiter:

Victor Christen

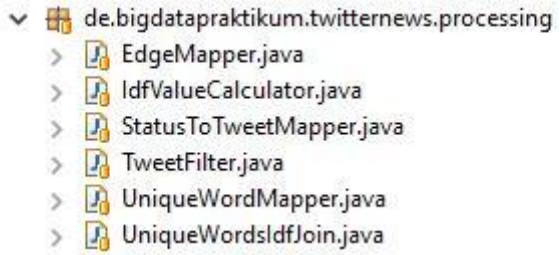
Konzeptioneller Entwurf

Generelles

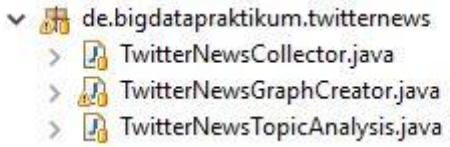
Als Framework für die Umsetzung wird, wie in der Aufgabenstellung gefordert, Apache Flink genutzt. Die Anwendung ist in unterschiedliche Packages gegliedert, welche in nachfolgender Abbildung zu sehen sind.



Die Packages orientieren sich grob am Flink Design und unterscheiden Sources von Processing. Im Processing liegen die Klassen, welche notwendige Datentransformationen durchführen:



Als Sources gibt es bei dem Programm die TwitterNewsSource, welche für das Laden der Tweets aus Twitter zuständig ist, sowie Tweets und Graph, welche als Repräsentation der geladenen Objekte dienen. Das übergeordnete Package enthält drei Klassen, die den drei Teilen der Aufgabenstellung entsprechen.



Das utils package enthält die Konfigurations-Klasse, die bestimmte Einstellungen und Konstanten enthält. In den folgenden Abschnitten werden bestimmte Designentscheidungen den Aufgabenteilen nach geordnet beschrieben.

Datenakquisition

Zur Datenakquisition wird die Bibliothek Twitter4j benutzt. Diese wird von der Klasse TwitterNewsSource verwendet und speichert einzelne Tweets als Status-Objekte. Im Zweiten Schritt werden die Status-Objekte zu eigenen Tweets-Objekten gemappt. Die Tweet-Objekte enthalten dabei nur noch die für die Aufgabenstellung relevanten Informationen. Zum Ende der Datenakquisition

werden die geladenen Daten noch im CSV-Format in eine Datei persistiert. Die Datei wird auch herangezogen, um zu ermitteln, welche Tweets noch geladen werden müssen. Es werden nur Tweets geladen, die noch nicht gespeichert wurden. Dazu wird das Änderungsdatum der Datei herangezogen, in der die Tweets gespeichert werden.

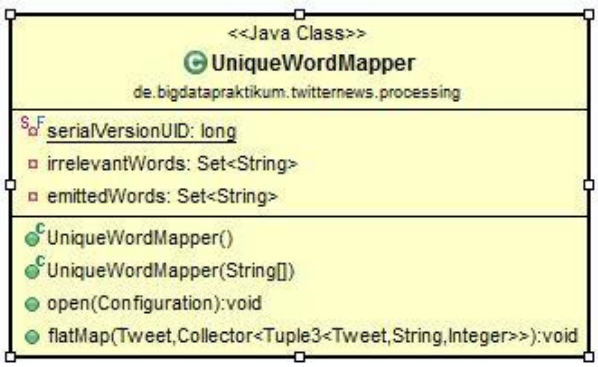
Diagramm der Klasse TwitterNewsSoruce:

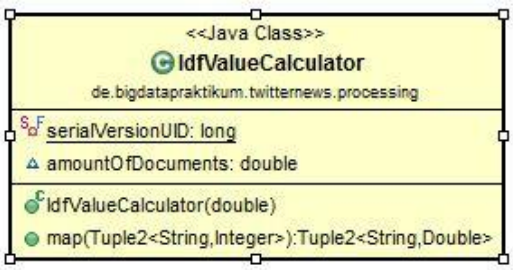
Preprocessing

In diesem Teil der Aufgabenstellung spielt das Processing eine größere Rolle. Zuerst werden die Daten aus dem vorherigem Schritt wieder eingelesen. Wichtig ist, dass die Akquisition unabhängig vom folgenden Programm stattfinden kann, damit es möglich ist Daten über einen längeren Zeitraum zu sammeln, ohne immer das ganze Programm mit allen Analysen laufen zu lassen.

Im ersten Schritt ist es nun wichtig, für jedes Wort das in irgendeinem Tweet verwendet wurde den IDF Wert zu berechnen. Dazu wird zuerst die Anzahl der gespeicherten Tweets ermittelt. Dann werden vom UniqueWordMapper alle doppelten oder irrelevanten Wörter aus dem Tweet ausgeschlossen. Um irrelevante Wörter zu erkennen wird eine Liste von deutschen Füllwörtern herangezogen, außerdem werden auch programmatische bestimmte Muster, wie z.B. URLS, gefiltert. Dann wird ermittelt, in wie vielen verschiedenen Tweets jedes einzelne Wort vorkommt und dann zusammen mit der Anzahl der Tweets der IDF-Wert für jedes Wort vom IdfValueCalculater berechnet.

Hier die UML-Diagramme für den UniqueWordMapper und den IdfValueCalculator:





Um nun für jeden Tweet die wichtigen Wörter herauszufinden wird durch den UniqueWordsIdfJoin ein JOIN zwischen den relevanten Wörtern und den einzelnen Wörtern in jedem Tweet gemacht. Übrig bleiben dann für jeden Tweet die zugehörigen relevanten Wörter. Ein Wort ist relevant, wenn es zu den 50 niedrigsten IDF-Werten gehört. Normalerweise zieht man den IDF-Wert heran und spricht einem hohen IDF-Wert auch eine hohe Bedeutung zu. Dies haben wir auch ausprobiert, bekamen aber sehr schlechte Ergebnisse, da es unglaublich viele Wörter gibt, die genau einmal vorkamen. Der Kookkurenzgraph dazu hatte keine Aussagekraft. Deshalb haben wir unwichtige Wörter (Füllwörter) im Voraus ausgeschlossen und dann die häufig auftretenden Worte für weitere Analysen genutzt. Wir haben uns gegen einen festen Thresholdwert entschieden, da die IDF-Werte je nach Anzahl der Dokumente stark variieren. Da wir unsere Analyse aber flexibel für verschiedene Zeiträum – und damit auch verschiedene Tweetmengen – halten wollten, haben wir uns dafür entschieden, die Top 50 Begriffe weiter zu untersuchen.

Die Menge von Tweets mit ihren wichtigen Wörtern wird dann an den TwitterNewsGraphCreator übergeben, der für die Erstellung des Kookkurenzgraphen zuständig ist.

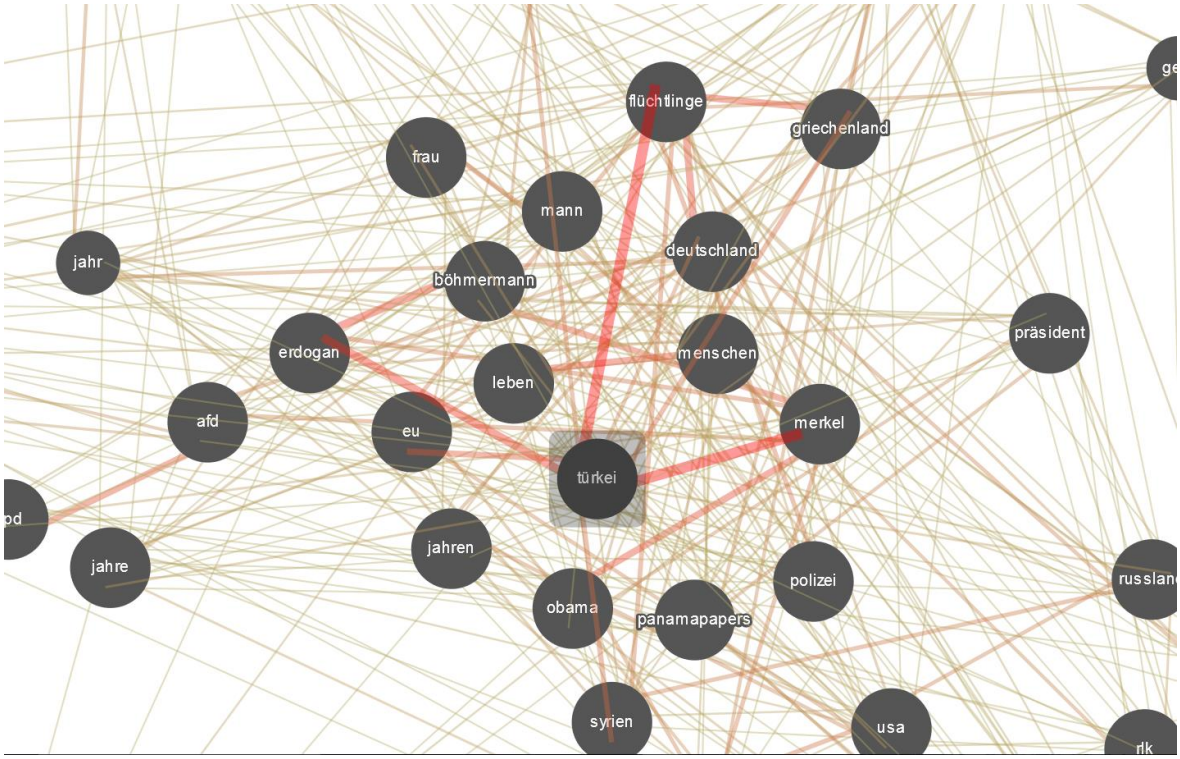
Kookkurenzgraph

Die wichtigste Processing-Klasse ist der EdgeMapper. Hier werden die Menge von Tweets mit den wichtigen Wörtern auf die Edges des Kookkurenzgraphen gemappt. Dabei werden die alle wichtigen Wörter des Tweets so miteinander kombiniert, dass es sich um eine Kombination ohne Wiederholung handelt. Die Reihenfolge spielt also keine Rolle, da jede Edge so gespeichert wird, dass der nach alphabetischer Reihenfolge kleinere Tweet als erster Knoten der Edge abgespeichert wird. Aus den Wörtern: „Erdogan“, „Boehmermann“, „Gedicht“, „Merkel“ werden also folgende 6 Tupel: („Boehmermann“, „Erdogan“), („Boehmermann“, „Gedicht“), („Boehmermann“, „Merkel“), („Erdogan“, „Gedicht“), („Erdogan“, „Merkel“), („Gedicht“, „Merkel“).

Die Edges werden dann noch aggregiert, sodass eine Kombination die öfter vorkommt auch einen höheren Edge-Value hat.

Um einen besseren Eindruck des Ergebnisses zu bekommen, haben wir die Dateien mittels Javascript und HTML visualisiert. Folgende Abbildung enthält einen Kookkurenzgraphen, der mittels unserer Analyse erstellt wurde. Die relevanten Themen der letzten Wochen lassen sich sehr gut erkennen.

Die Abbildung auf der nächsten Seite zeigt den visualisierten Kookkurenzgraphen.


flüchtlinge
griechenland
frau
mann
jahr
deutschland
böhmermann
präsident
erdogan
menschen
leben
afd
eu
merkel
türkei
jahren
jahre
polizei
obama
panamapapers
syrien
usa