Freie Universität Berlin
Institut für Informatik
Datenbanksysteme
Sommersemester 2017
Bearbeiter: Felix Binder, Nicolas Höcker, Armin Weber

Projektdokumentation

1. Projektziel

Ziel des Projekts ist es, eine „Projekt-Web-Anwendung“ zu erstellen, mit deren Hilfe der Datensatz „american-election-tweets“ analysiert werden kann. Analysieren heißt hier, dass die Anwendung in der Lage sein soll, das Beziehungsgeflecht zwischen den in den Tweets verwendeten Hashtags grafisch darzustellen und mittels benutzerdefinierter Anfragen weitere Informationen aus den Daten zu gewinnen. So soll es unter anderem möglich sein, die am meisten verwendeten Hashtags zu identifizieren, Aussagen über deren paarweise gemeinsames Auftreten zu treffen und die Wichtigkeit einzelner Tweets zu bewerten. Auch Auswertungen dessen, wie sich die einzelnen Punkte über die Zeit hinweg verändert haben, sollen möglich sein.

2. Team

Felix Binder: Studiert Philosophie und Informatik (60 LP) gegen Ende seines Bachelorstudiums.

Nicolas Höcker: Studiert Informatik im 4. Bachelorsemester.

Armin Weber: Studiert Informatik im vierten Semester.

3. Explorative Datenanalyse

Der Datensatz besteht aus 6126 Tweets, die zwischen dem 5. Januar 2016 und dem 28. September 2016 über die Twitter-Konten *realDonaldTrump* und *HillaryClinton* abgesetzt, d. h. von diesen selbst verfasst oder retweetet wurden. Das jeweilige Twitter-Konto findet sich dabei in der ersten Spalte des Datensatzes („handle“); die Information, ob es sich um einen Retweet handelt, in der dritten Spalte („is_retweet“), in der vierten bei Retweets der ursprüngliche Autor („original_author“). Der Text selbst – und mit ihm alle Hashtags – befindet sich in der zweiten Spalte.

Darüber hinaus enthält der Datensatz Informationen darüber, ob der Text ein Zitat ist und also auf eine andere Quelle verweist (siebte Spalte, „is_quote_status“, mit der Quelle in der neunten Spalte, „source_url“) sowie dabei evtl. abgekürzt wurde (zehnte Spalte, „truncated“), ob der jeweilige Tweet eine Antwort auf einen anderen Tweet darstellt (sechste Spalte, „in_reply_to_screen_name“), wie oft er retweetet wurde und wie oft favorisiert (Spalten 7 und 8).

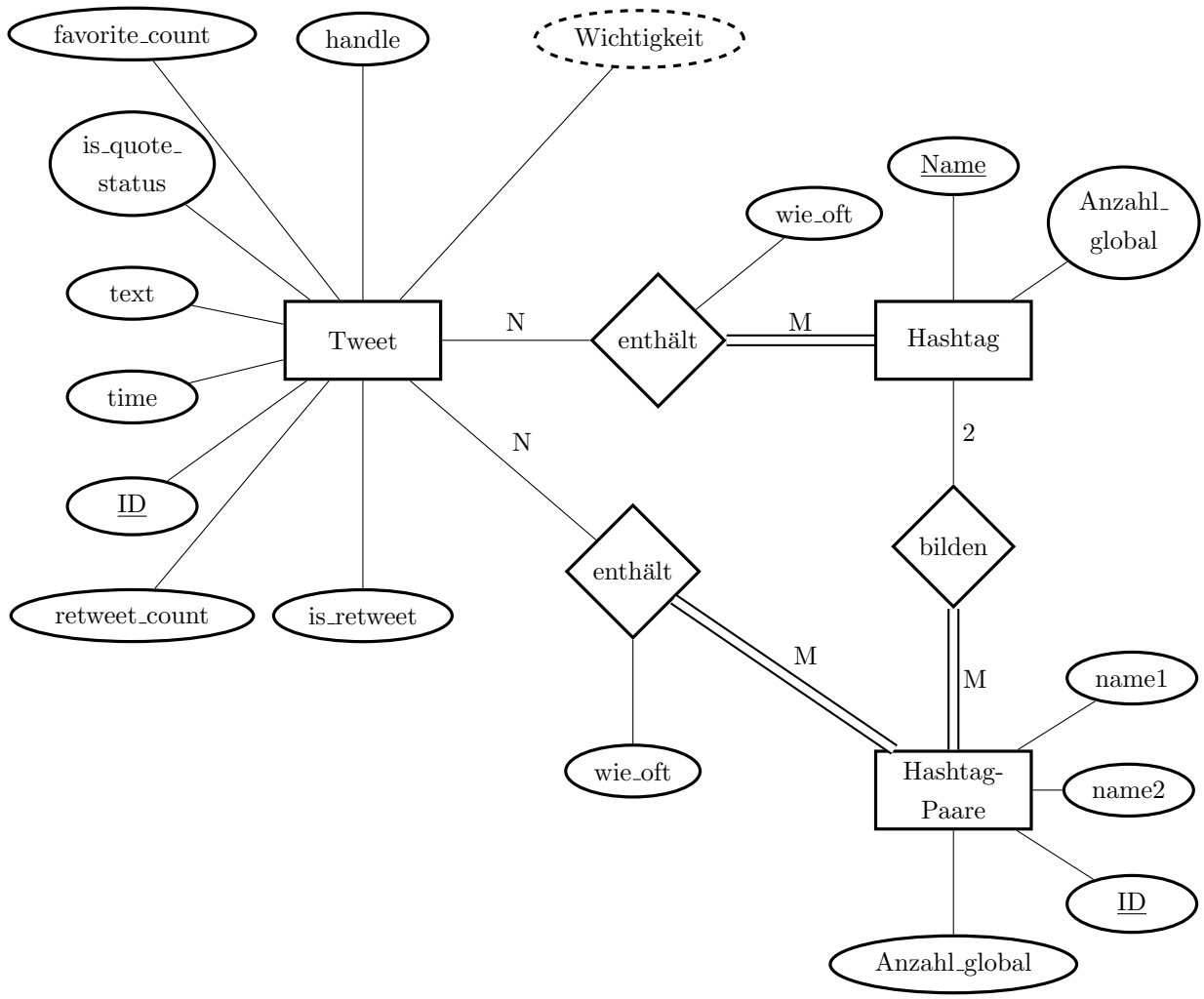
Dem ersten Augenschein nach sind für uns von Bedeutung:

a) die Spalte Text, um daraus die Hashtags zu extrahieren;
b) die Spalte mit dem Datum und der Uhrzeit, um das unterschiedlich starke Vorkommen von Hashtags zu verschiedenen Zeitpunkten nachvollziehen zu können;
c) die Spalten zu Retweets und Favorisierungen, weil sie Indikatoren für die Wichtigkeit von Tweets sind.

Die weiteren Spalten scheinen auf den ersten Blick für unsere Zwecke vernachlässigbar zu sein.

4. ER-Modellierung

Das ER-Modell für das Projekt sieht wie folgt aus:



Dieses Modell enthält augenblicklich noch mehr Informationen, als für die Beantwortung der in Abschnitt 1 genannten Fragen nötig wäre. So wäre es eigentlich unnötig abzuspeichern, ob ein Tweet ein Retweet oder ein Zitat ist. Allerdings wollen wir die Option offenhalten, in der fertigen Anwendung zu untersuchen, ob es signifikante Unterschiede hinsichtlich der Wichtigkeit der Tweets der verschiedenen Arten gibt. Da wir solche Unterschiede nicht ausschließen können, scheint uns das Weglassen solcher Informationen zu diesem frühen Zeitpunkt nicht angeraten.

Davon abgesehen war unser Anliegen, immer die naheliegenden Entscheidungen zu treffen: etwa Hashtag-Paare aus Hashtags zusammenzusetzen und in den Relationen alle Informationen zu speichern, die später abrufbar sein sollen, etwa wie oft Hashtags in einzelnen Tweets vorkommen. Dort, wo es sinnvoll erschien, haben wir als Schlüssel außerdem eigenständige IDs ergänzt, um keine zu komplexen Schlüssel verwenden zu müssen (wie es bei den Tweets sonst etwa der Fall wäre).

Außerdem haben wir uns nach einigem Hin und Her dazu entschlossen, den Autor eines

Tweets als Attribut handle in das Modell zu integrieren statt als eigene Entität. Hätten wir ihn als eigene Entität modelliert, die zu Tweets in den Relationen „schreibt“ und „retweetet“ steht, so hätten wir eine Tabelle mit nur einem einzigen Attribut (eben „handle“), wohingegen alle Attribute der Relationen zu „Tweet“ gezogen hätten werden können, da es sich um 1:N-Relationen gehandelt hätte. Damit hätte man für Abfragen nach dem Autor eines Tweets eigens mehrere Tabellen mittels JOIN verbinden müssen, was gerade bei Abfragen von den Hashtags her reichlich komplex geworden wäre, ohne dass das Ausgliedern des Autors als Entität irgendeinen Vorteil zu versprechen schien.

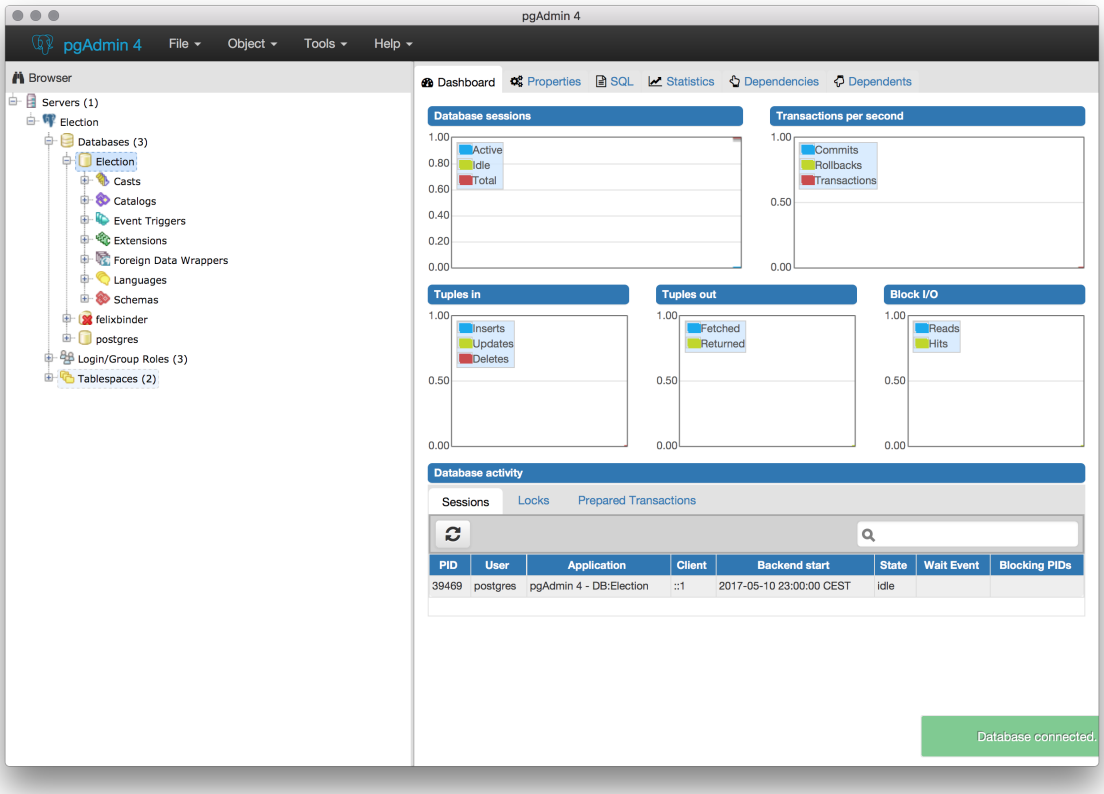
5. Relationales Modell

TWEET(ID, handle, text, time, is_retweet, is_quote_status, retweet_count, favorite_count)
HASHTAG(Name, Anzahl_global)
T_ENTH_H(Tweet_ID, H_Name, wie_oft)
HASHTAG_PAARE(ID, Anzahl_global, name1, name2)
HASHTAGS_BILDEN_HP(Hash_Name, HP_ID)
T_ENTH_HP(Tweet_ID, Hashpaar_ID, wie_oft)

Grundsätzlich haben wir uns bei der Erarbeitung des relationalen Modells an die systematische Umwandlung aus der Vorlesung bzw. den Tutorien gehalten. Neben der korrekten formalen Übertragung und der daraus resultierenden funktionalen Bedeutung der Tabellen waren uns eine einleuchtende Benennung und eine gute Strukturierung wichtig, die Dritten das Verständnis erleichtert. Die Benennung der Tabellen haben wir so gestaltet, dass man aus den Namen möglichst sofort die inhaltliche Bedeutung ablesen kann. Dies war besonders für die beiden „enthält“-Relationen aus dem ER-Modell wichtig. Auch die zusammengesetzten Primärschlüssel der Relationstabellen haben wir klar benannt. Bei allen Relationstabellen haben wir uns für einen zusammengesetzten Primärschlüssel aus den verbundenen Entitätstabellen und damit auch überhaupt für die Beibehaltung derselben entschieden. Die Möglichkeit, die Relationen direkt in eine Entität auszulagern, erschien uns zu unstrukturiert und schlechter erfassbar für Dritte, die unsere Datenbank schnell verstehen möchten.

6. Datenbank

Wir haben eine Datenbank *Election* erstellt.



Der Code dafür war:

```
CREATE DATABASE "Election"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    CONNECTION LIMIT = -1;
```

7. Datenbankschema erstellen

https://github.com/felixbinder/FU_DBS_Projekt/blob/master/election.sql

8. Datenbereinigung

https://github.com/felixbinder/FU_DBS_Projekt/blob/master/Convert.java

9. Datenimport

https://github.com/felixbinder/FU_DBS_Projekt/blob/master/Insert.java

10. Webserver

Wir haben einen Apache2-Server unter Ubuntu aufgesetzt. Dazu nutzten wir sudo apt-get install apache2. Der Server wird mit sudo service apache2 start gestartet. Die Konfiguration haben wir unseren Bedürfnissen angepasst.

11. Metrik für Ähnlichkeit von Hashtags

Wir wählen eine Metrik für die Ähnlichkeit, die auf zeitlicher Ähnlichkeit im Auftreten der Tweets beruht. Gedanke dahinter ist, dass zeitlich aufeinanderfolgende Tweets tendentiell thematisch zusammenhängen. Die Metrik sucht für jeden Tweet A den nächsten Tweet B und berechnet den zeitlichen Abstand. Die zeitlichen Abstände werden quadriert, mit 1 geteilt und summiert und der Durchschnitt gebildet. Somit bildet die Metrik die durchschnittliche quadratische zeitliche Distanz zu einem Auftreten des Hashtags. Die Metrik wird in beide Rechnungen berechnet, um eine symmetrische Metrik zu schaffen.

```
[...]
    for htA in hashtags:
        for htB in hashtags:
            print("Berechne: "+str(htA[0])+" und "+str(htB[0]))
        #Berechne Abstand htA und htB
            final_ae = aehnlichkeit(htA,htB) + aehnlichkeit(htA,htB)
            cursor.execute("INSERT INTO hashtags_aehnlichkeit(name1,name2,
                                                    aehnlichkeit) VALUES('"+
                                                    str(htA[0])+"','"+str(htB
                                                    [0])+"',"+str(final_ae)+"
                                                    );")
   [...]
def aehnlichkeit(htA, htB):
    aehnlichkeit = 0
    count = 0
    cursor.execute("SELECT time FROM tweet, t_enth_h WHERE tweet.ID =
                                            t_enth_h.tweet_id AND h_name = '"
                                             + str(htA[0]) + "';")
    atweets = cursor.fetchall() #[alle Tweets, die Hashtag A enthalten]
    cursor.execute("SELECT time FROM tweet, t_enth_h WHERE tweet.ID =
                                            t_enth_h.tweet_id AND h_name = '"
                                             + str(htB[0]) + "';")
    btweets = cursor.fetchall() #[alle Tweets, die Hashtag B enthalten]
    for at in atweets:
        k=0
        distance = []
        for bt in btweets:
            distance.append(float(abs((at[0] - bt[0]).total_seconds())))
```

```
        md = min(distance)
        if md == 0:
            aehnlichkeit+100 #Wert waehlen!
        else:
            aehnlichkeit += 1000/min(distance)
    return aehnlichkeit
```

Quelldatei: https://github.com/felixbinder/FU_DBS_Projekt/blob/master/metrik_zeit.py

12. K-Means

Jeder Hashtags hat nach der Berechnung der Ähnlichkeitsmetrik einen Ähnlichkeitswert zu allen anderen l Hashtags. Jeder Hashtag hat somit einen l-dimensionalen Vektor. Auf Basis dieses Vektors (die zusammen einen Vektorraum aufspannen), der die Ähnlichkeit zu allen anderen Hashtags beschreibt, führen wir k-means aus.
Die Ergebnisse von K-Means werden bei Aufgabe 2.3 visualisiert. Dafür fügt das Skript zur Berechnung von K-Means für jeden Hashtag in die Datenbank zwei Werte ein: erstens den zugehörigen Cluster und zweitens markieren wir die k Hashtags, die für ihren Cluster am repräsentativsten sind. Die berechnen wir, in dem wir für jeden Centroid den Hashtag berechnen, dessen Position im oben beschriebenen Vektorraum am nächsten (euklidische Distanz) zu dem Centroid ist. Dieser repräsentiert dann den Cluster als dessen typischster Vertreter. Der Code findet sich hier: https://github.com/felixbinder/FU_DBS_Projekt/blob/master/Clusterfindung.py

13. Visualisierung des Hashtagnetzwerks

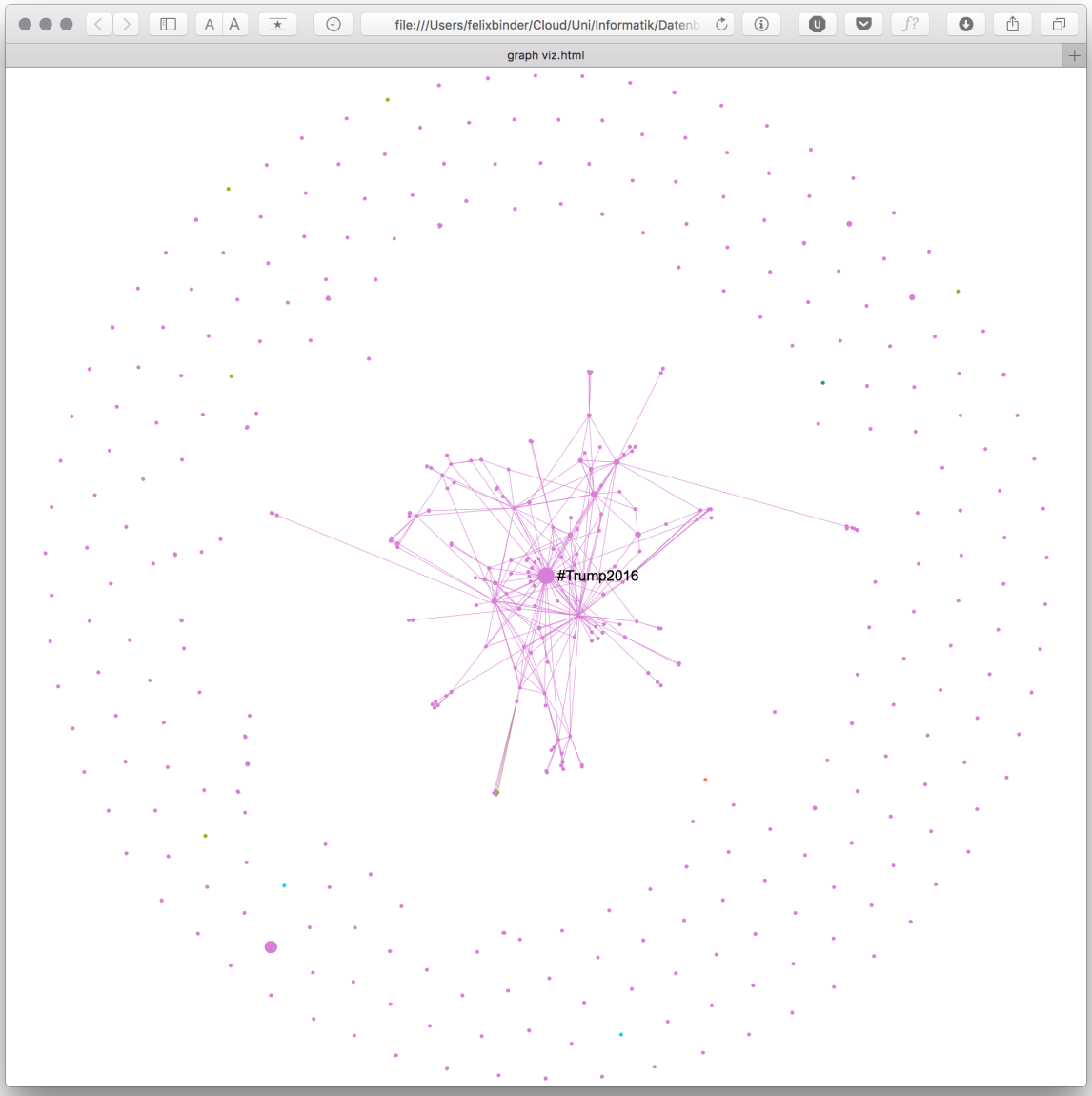
13..1 Aufbereitung des Graphens

Nachdem die Daten aus der Datenbank exportiert wurden und manuell in Excel gereinigt wurden, wird mithilfe von Gephi ein Graph erstellt. Die Exceltabelle findet sich hier: https://github.com/felixbinder/FU_DBS_Projekt/blob/master/Graph-Rohdaten.xlsx, die Graph-Datei hier: https://github.com/felixbinder/FU_DBS_Projekt/blob/master/graph_final.gexf

13..2 Visualisierung

Die Daten werden mit sigma.js visualisiert. Dabei entspricht die Größe des Knoten der absoluten Zahl des Auftretens des entsprechenden Hashtags. Die Farbe der Hashtags entspricht deren Zugehörigkeit zu einem k-means-Clusters. Der repräsentativste Hashtag eines Cluster wird mit einem Diamanten markiert. Der Graph wird mit dem AtlasForce2 Algorithmus visualisert.

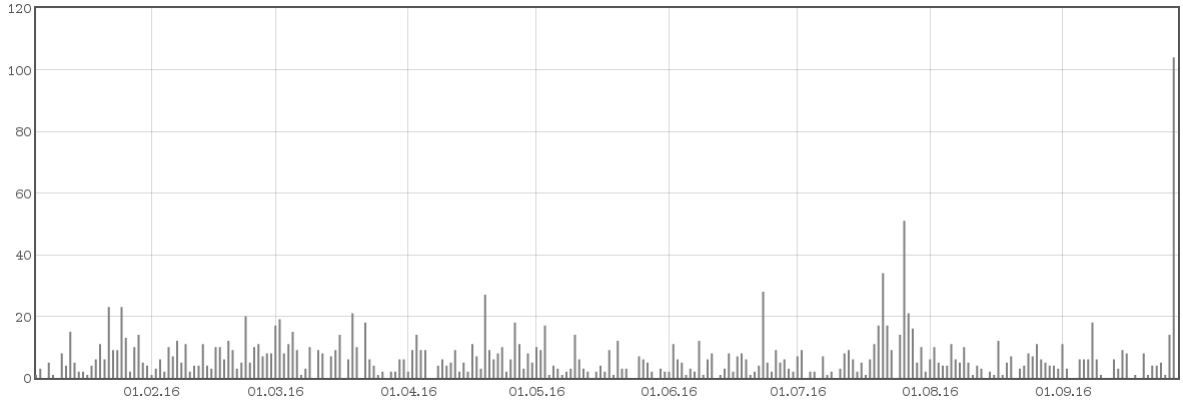
Dieser Algorithmus ist ein physik-basierter Layout-Algorithmus. Er verwendet die Knoten zwischen Kanten als Federn, deren Gewicht (hier: Ähnlichkeit) deren Stärke beschreibt. Beim Betrachten der Visualisierung kann beobachtet werden, wie die Knoten sich in dem Graphen sich in den ersten Sekunden verschieben, bis schließlich ein Gleichgewicht eingestellt hat und der Graph so gerückt wurde, dass die Knoten nach ihrer Ähnlichkeit (via Kantengewicht) geordnet wurden (wobei nicht zwangsläufigerweise eine optimale Lösung erreicht wird). Die Visualisierung liegt in der Datei `FU_DBS_Projekt/sigma.js/graph viz.html`.



Da die Datei auf Bibliotheken von sigma.js zugreift, muss die Ordnerstruktur gewahrt bleiben. Hier die entsprechende Orderstruktur als .zip: `https://github.com/felixbinder/FU_DBS_Projekt/blob/master/sigmajs.zip` (es ist zu beachten, dass die Graph-Datei im selben Verzeichnis wie der Ordner liegen muss).

14. Visualisierung aller Hashtags

Die Häufigkeit des Autretens aller Hashtags insgesamt über die Zeit hinweg haben wir mittels PHP und JavaScript visualisiert. Der Quellcode ist hier zu finden: `https://github.com/felixbinder/FU_DBS_Projekt/blob/master/a2_4.php` Die Visualisierung selbst sieht wie folgt aus:



15. Visualisierung einzelner Hashtags

Die Häufigkeit des Autretens einzelner Hashtags über die Zeit hinweg haben wir mittels PHP und JavaScript visualisiert. Der Quellcode ist hier zu finden: `https://github.com/felixbinder/FU_DBS_Projekt/blob/master/a2_5.php` Die Visualisierung selbst sieht wie folgt aus (zwei zufällig ausgewählte Beispiele):

Hashtag: #1 Anzeigen

Übersicht Auftreten des Hashtags #MakeAmericaGreatAgain