1 Einleitung

Soziale Netzwerke wie Facebook, Instagram und Twitter sind in den letzten Jahren sehr stark gewachsen und haben somit einen großen Platz in unserer Gesellschaft eingenommen. Jede dieser Plattformen hat täglich mehrere Hunderte Millionen an aktiven Nutzern. Diese Nutzer kreieren oder konsumieren tagtäglich "Content" auf diesen Plattformen. Auch wenn man selber keinen "Content" erzeugt, interagiert man immer mit den anderen Nutzern in Form von Likes, Kommentaren oder impressionen also das erreichen/sehen des Posts.

Dadurch sind die Sozialen Netzwerke auch ein großens Gebiet in der Forschung geworden, da man sonst nirgendswo auf so viel Daten von Usern stößt. Es ist immer interressanter geworden, zu verstehen wie die Kommunikation auf solchen Plattformen in der Öffentlichkeit vonstattengeht. Denn man kann somit untersuchen und verstehen wie Menschen online mit anderen Menschen kommunizieren. Des weiteren ist es möglich Konversationen und Diskussionen über bestimmte Themen zu analysieren. Es ist möglich das Verhalten der Menschen auf bestimmte Themen/Informationen zu verstehen und nachzuvollziehen wie diese Menschen damit umgehen. Und es ist möglich nachzuvollziehen wie sich Themen/Informationen in Sozialen Netzwerken ausbreiten. Wodurch sich dann ganze Netzwerke an User und Antworten auf die Themen/Informationen bilden.

Dadurch das Twitter eine sehr beliebte Plattform für den Themen-Austausch ist, beschäftigt sich diese Arbeit mit der "Microblogging"-Plattform Twitter. Denn auf Twitter finden Konversation nicht nur zwischen einzelnen Usern statt, sondern werden durch Öffentlich sichtabaren "Trends" von Millionen an Usern geführt. Deswegen ist es hier besonders spannend mit diesen Konversationen zu arbeiten, da hier besonders viele User aufeinander prallen und an den Konversationen teilnehmen können. Desweiteren bietet Twitter eine API Schnittstelle an, welches das arbeiten mit den Konversation überhaupt erst möglich macht.

Mit der Twitter API wird in dieser Arbeit ein verfahren entwickelt, ganze Konversationen von Twitter abzugreifen. Dazu müssen möglichst große Konversationen gefunden werden, diese müssen extrahiert werden und in einer analysierbaren Form gespeichert werden damit diese Konversationen in Zukunft weiter verwertbar sind. Dies gilt es zu schaffen, ohne die Twitter API an ihr limit zu bringen, denn diese API ist für Drittsysteme zur begrenzt verfügbar. Diese begrenzte Verfügbarkeit ist dadruch zu stande gekommen, das Soziale Medien in de letztn Jahren immer mehr unter Druck der Öffentlichkeit stehen um Nutzer Daten vor Drittsystemen zu schützen. Desweitere sind Nutzer Daten für diesew Plattformen das höchste Gut, da sie damit auch ihr Geld verdienen. Deswegen stellen Sie diese Dienste nur begrenzt zur Verfügung, was ein bestimmtes Abfrage limit in einem bestimmten Zeitraum bedeutet.

1.1 Problembeschreibung

Um von Twitter ganze Konversationen zu extrahieren, und diese strukturiert darzustellen, gilt es einige Probleme auszumerzen. Das erste Problem was es zu lösen gilt, ist es Konversationen zu finden an denen möglichst viele Interaktionen stattfinden. Danach gilt es einen Algorithmus zu entwickeln und zu implementieren, welcher diese Konversationen abgreift. Dabei beinhaltet eine Konversation den "Haupt-" Tweet, welcher der Ankerpunkt der ganzen Konversation ist. Auf diesen "Haupt-" Tweet können unbegrenzt viele Antworten von unbegrenzt vielen Usern stattfinden. Dieser "Haupt-" Tweet und die Antworten bilden dann die gesamte Konversation.

Die Konversationen, also der "Haupt-" Tweet und die "Antwort-" Tweets werden durch die Twitter-API abgegriffen. Die nächste Hürde die es zu überwinden gilt, ist die Limitierung der Twitter-API. Denn diese Limitierung beeinhaltet zum Beispiel die Regel, das innerhalb von 15 Minuten nur 300 Tweets abgefragt werden können. Dieses Limit ist bei großen Konversationen natürlich schnell erreicht. Deswegen gilt es in dieser Arbeit trotz der Einschränkungen ein verfahren zu finden, welches das abgreifen von den großen Konversationen möglich macht. Dabei gilt es außerdem zu berücksichtigen, das Twitter über die Twitter-API nicht direkt die gesamte Konversation ausliefert, sondern lediglich eine Id ausgibt, zu welcher Konversation ein Tweet gehört.

Dadurch das man nicht direkt an die gesamte Konversation durch die Twitter-API kommt, sondern nur an die Tweets mit einer Id, zu welcher Konversation diese Tweets gehören, gilt es ein Weg zu finden, um dennoch die gesamte Konversation, so wie diese auf Twitter stattgefunden hat, auch wieder darzustellen.

Des Weiteren sind nach dem einmaligen Abgreifen der Daten, die Konversationen nicht beendet. Also muss in dem Verfahren berücksichtigt werden, das Konversation im laufe der Zeit wachsen können. Somit müssen die Konversationen nicht nur einmalig, initial abgegriffen und wieder strukturiert dargestellt werden, sondern Sie müssen auch erweiterbar sein und erweiterbar bleiben. Deswegen bietet es sich an die Konversationen nach dem Sie abstrahiert wurden sind, als Graphen darzustellen. Denn somit ist die Konversation geordnet und immer erweiterbar sowohl in der Breite, als auch in der Tiefe.

1.2 Aufbau der Arbeit

In Kapitel 2 werden alle technischen und fachlichen sowie Konzepte und Definitionen vorgestellt.

In Kapitel 3 wird der Aufbau und die durchführung des Algorithmusses nähergebracht, welche die Lösung auf die Probkembeschreibung wiederspiegelt.

In Kapitel 4 wird der Datensatz welcher sich aus der Visualisierung der gesammelten Daten, welche sich aus dem Algorithmus aus Kapitel 3 ergeben analysiert.

In Kapitel 5 wird die Analyse aus Kapitel 4 mit anderen Verfahren verglichen und bewertet.

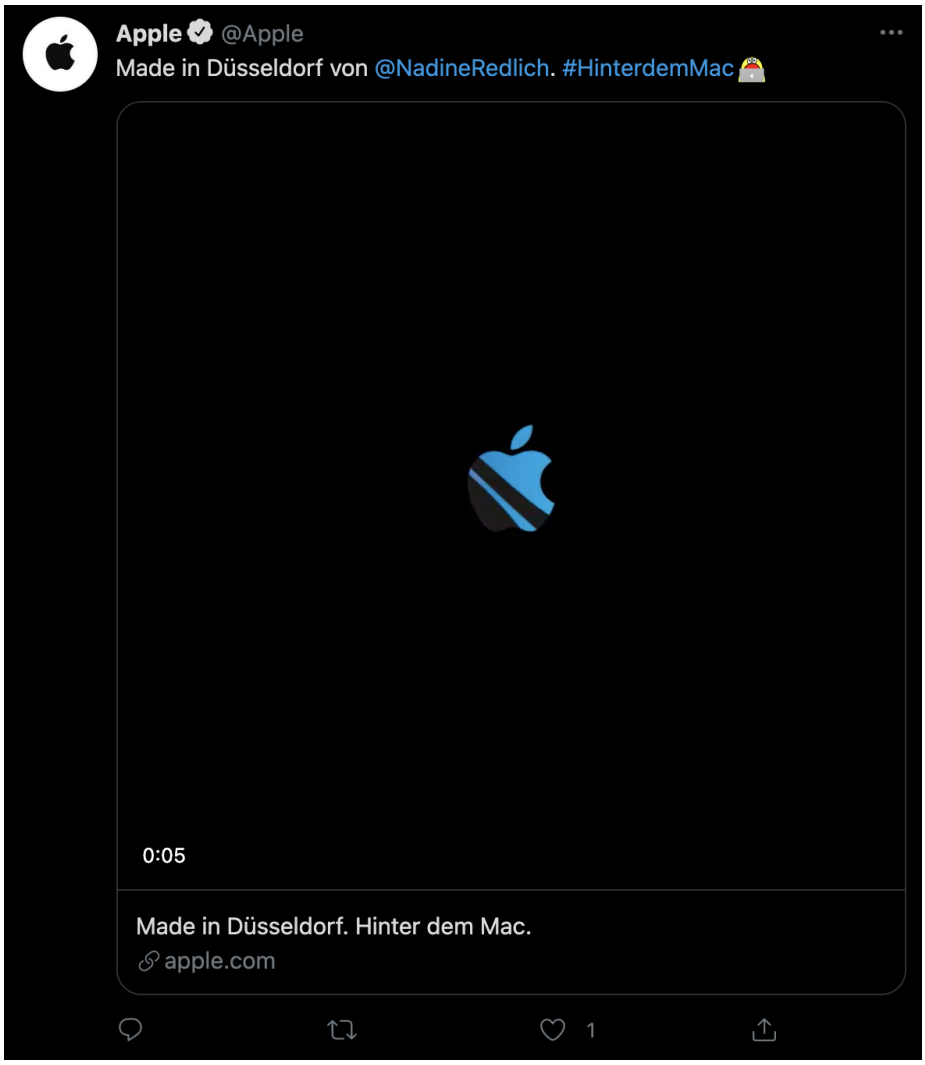
In Kapitel 6 gibt es eine Zusammenfassung und ein Fazit zu den vorherigen Kapiteln. Desweiteren wird es einen Ausblick geben wozu die Datensätze verwendet werden können und wo es gegebenenfalls noch Verbesserungspotential gibt und welche Probleme für die Zukunft noch offen bleiben.

2 Grundlagen

In diesem Kapitel, werden die wichtiges Komponenten und Definitionen die zum verstehen des Algorithmus und der Analyse des Datensatzes gebraucht werden. Außerdem wird die technische Implementierung erläutert um das Zusammenspiel der Komponenten zu verstehen.

2.1 Twitter

Twitter ist ein Mikroblogging Dienst, der sich darauf konzentriert hat wichtige Kurznachrichten durch Ihre User zu verbreiten. Diese Kurznachrichten, werden meist über "Hashtags" gruppiert. User des Dienstes Twitter haben einen Account welcher aus einem Namen, der Herrkunft und den Interressen des Users besteht. Außerdem können User anderen Usern "folgen" und genauso können andere User einem selbst "folgen". Diese "folgen" Funktion hat den Vorteil, das der User auf seiner "Timeline" also auf seiner Startseite alle Interaktionen der User sieht, denen er folgt. Diese Interaktionen, können aus Tweets, Antworten auf Tweets, Likes oder Retweets bestehen. Ein Tweet, also eine Kurznachricht welche von einem User verfasst wird, kann bis zu 140 Zeichen enthalten. Des weiteren können die Tweets um "Hashtags" welche mit einer Raute vorrangehen und mit Erwähnungen an anderen User welche mit einem "@" vorrangehen erweitert werde. Die "Hashtag" Funktionen bietet den Vorteil, dass dieser Tweet von anderen Nutzern unter diesem Thema öffentlich einsehbar ist. Die Erwähnung andere User, bei Twitter auch "Mention" genannten, hat den Vorteil das dieser Tweet an diese Person direkt gerichtet wird und die erwähnte Person eine Benachrichtigung erhält, dass ein Tweet an Sie gerichtet wurden ist.



Beispiel für ein Tweet mit Erwähnung eines Users und Themen Gruppierung: Ein Tweet des Users "Apple", welcher sich direkt an de Nutzer "NadineRedlich" richtet und unter dem "Hashtag" "HinterdemMac" zu finden ist. Desweiteren ist neben dem "Herz" Symbole eine "1" zu sehen, was bedeutet das dieser Tweet von einer Person "geliked" wurden ist."

Des Weiteren haben User die Funktion Tweets zu "Retweeted", was eine Art Zitierfunktion von Tweets darstellt.

Ein "Tweet-" Zitat des Users "HHU", welcher somit den Tweet für seine "Follower" teilt und verbreitet.

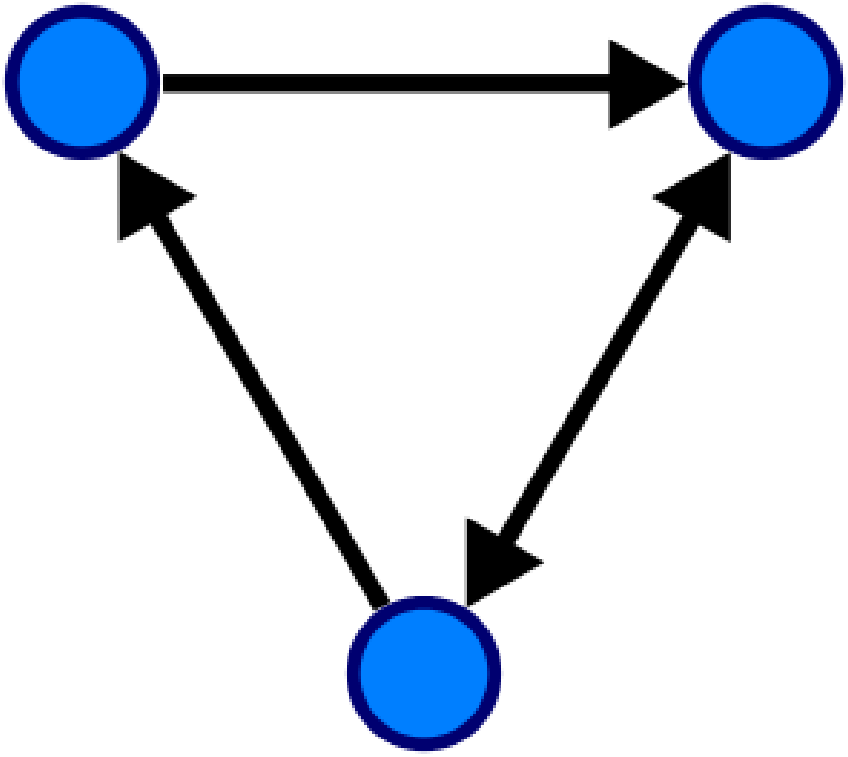
Als letztes, gibt es noch die Möglicheit auf einen Tweet zu Antwortet. Somit können User auf den Tweet reagieren und ihre Meinung dazu abgeben. Dadurch entstehen teilweise sehr lange und interressante Diskussionen, welche interressierte User nicht nur mitlesen können sondern wenn Sie möchten, können diese auch jederzeit mitdiskutieren. Dadurch das ein User an einer Konversation teilgenommen hat, also eine Antwort auf einen Tweet verfasst hat, sehen seine "Follower" diese Interaktion und haben somit auch die Chance die Konversation mitzuverfolgen und teilzunehmen.

Ein Tweet des Users "catalinmpit", welcher 3 Antworten von anderen Usern beeinhaltet.

Und auf diese Form der Interaktionen, konzentriert sich diese Arbeit. Diese Tweets mit Antworten, welche ganze Konversationen darstellen, werden in dieser Arbeit analysiert, extrahiert und rekonstruiert.

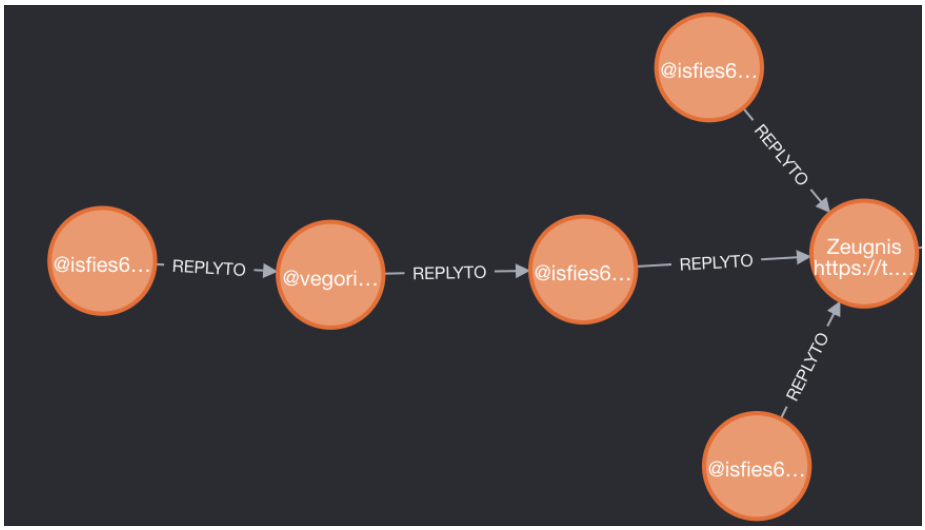
2.2 Graphen

Ein Graph G ist ein Tupel bestehend aus G = (V, E) mit einer nicht leeren Knotenmenge V und einer Kantenmenge E. Visualisiert wird der Graph G, dadurch das die Knoten V als Kreis abgebildet werden und die Kanten E als Verbindungslinien zwischen den Knoten V.



Beispiel Abbildung eines Graphen mit 3 Knoten und 3 Verbindungslinien zwischen diesen Knoten.

In dieser Arbeit werden die Twitter Konversationen wie folgt im Graphen dargestellt: Ein Tweet ist ein Knoten aus der Menge V und die Verbindungslinie aus der Menge E zeigt an, welcher Tweet worauf antwortet.



Beispiel einer Twitter Konversations Darstellung als Graph.

2.3 API - Schnittstellen

Eine API - Application Programming Interface ist eine Anwendungsschnittstelle, welche Daten und Funktionen für Drittsysteme zur Verfügung stellt. In dieser Arbeit wird auf die Twitter API zugegriffen, welche Tweets, User etc. bereitstellt. Diese Funktionen bzw. Daten sind für diese Arbeit essentiell um die benötigten Konversationen extrahieren zu können.

2.4 Docker

Docker ist eine Anwendung, welche Dienste wie Datenbanke oder ähnliche Services in eine Container-Umgebung zum laufen bringt. Somit wird Docker in dieser Arbeit verwendet um die Datenbank zu verwalten, in welcher die extrahierten Tweets gespeichert werden.

2.5 Neo4j

Neo4j ist eine Open-Source-Graphenbank. Die anstelle von relationalen Datenbanken keine Tabelle sondern Graphen anlegt. Dieser Typ von Datenbank bietet sich in dieser Arbeit an, da die Tweets als Graph dargestellt werden.

2.6 Java und Spring Boot

In dieser Arbeit wird eine Spring Boot Applikation verwendet, welche als Programmiersprache Java im Einsatz hat. Hier kommt Spring Boot zum Einsatz, da

dieses Framework die Konfigurationen zur TwitterAPI und auch zur Datenbank vereinfacht.

2.7 Technische Implementierung

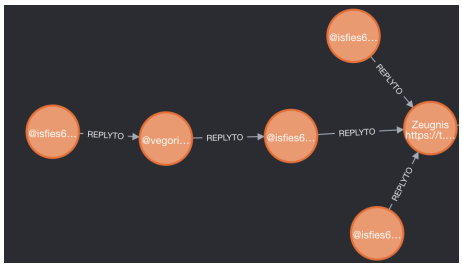
Mit Hilfe von Docker wird zunächst die Datenbank Neo4j gestartet. In der Spring Boot Applikation wird dann eine Verbindung zur Twitter API und eine Verbindung zur Neo4j Datenbank aufgebaut. Durch die Java Library "Twitter4j" lassen sich dann vereinfacht die Tweets, User etc. von der Twitter-API extrahieren. Diese Tweets werden dann in die Graph-Datenbank abgelegt.

3 Ein Algorithmus zur Lösung des Problems

In diesem Kapitel wird der Algorithmus vorgestellt, welcher sich um die für diese Arbeit relevaten Twitter Konversationen kümmert. Dieser Algorithmus versucht möglichst große Konversationen auf der "Microblogging-Plattform" Twitter zu finden. Daraufhin ist das Ziel des Algorithmus, diese Konversationen über die Twitter API zu extrahieren. Wenn diese Twitter Konversations-Daten über die Twitter API extrahiert wurden, ist der Algorithmus dafür verantwortlich diese Konversation zu rekonstruieren und als Graph in der Neo4j Datenbank zu speichern. Dabei soll die Konversation so dargestellt werden, wie sie auf Twitter stattgefunden hat. Desweiteren müssen diese "Koversations-Graphen" erweitert werden, wenn auf Twitter die Konversation erweitert wurden ist.

Erläuterung anhand eines kleinen Beispiels:

Start: Ausgangssituation der Twitter Konversation



Ziel: Rekonstruktion der Twitter Konversation als Graph

3.1 Beschreibung der Vorgehensweise

Somit gilt als Hauptaufgabe des Algorithmus, Konversations-Daten zu sammeln und diese als Graph zu visualisieren. Dazu müssen als erstes brauchbare Konversationen auf Twitter gefunden werden. Um brauchbare Konversationen auf Twitter zu finden, geht der Algorithmus erstmal die "Trends" durch. "Trends" sind Hashtags, also gruppierte Themen, die zu diesem Zeitpunkt sehr viel Interaktion haben. Auf dem ausgewählten "Trend", wird dann der Tweet rausgesucht, welcher die höchste Interaktion hat. Hier kann jedoch was die Interaktion angeht, nur auf die Retweet und Like Verhältnisse geschaut werden, da die Twitter API keine Anzahl an "Replies", also Antworten auf Tweets bereitstellt. Wenn nun ein Tweet gefunden wurden ist, welcher mit hoher wahrscheinlichkeit eine ganze Konversation abbildet, muss dieser als nächstes extrahiert werden. Da die Twitter API jedoch keine "Replies", also Antworten auf Tweets zur Verfügung stellt, musste hier ein Algorithmus entwurfen werden, welcher die Konversationen extrahiert. Dazu wird auf der Twitter API, der "Haupt-" Tweet also der Ursprung der Konversation aufgerufen. Auf diesem "Haupt-" Tweet lassen sich dann alle stattgefundenen Tweets herausfiltern. Jedoch nicht sortiert und strukturiert, wie die Konversation stattgefunden hat. Deswegen müssen diese Tweets rund um den "Haupt-" Tweet über das Feld "getInReplyToStatusId()", welches die Twitter API zur Verfügung stellt, abgeglichen werden, um zu wissen das diese Tweets Antworten auf den Ursprung sind. Diese Antworten werden in einer Liste gespeichert. Wenn alle Antworten auf den "Haupt-" Tweet gefunden wurden sind, wird der "Haupt-" Tweet als Knoten in die Graph-Datenbank abgelegt. Daraufhin wird die über die Antwort-Liste iterriert und die Antwort Elemente in die Graph-Datenbank gespeichert, inklusive Verbindungslinie zum "Haupt-" Tweet. Dieser Vorgang wird rekursiv ausgeführt, um auf den Antwort Tweets, wieder alle Antworten zu finden. Diese rekursion wird so lange ausgeführt, bis zu keiner Antwort mehr eine weitere Antwort gefunden wird. Danach werden also bisherigen Konversationen, welche sich in der Graph-Datenbank befinden erweitert. Bezieungsweise werden die in der Datenbank gespeicherten Tweets, mit den dazugehörigen Konversationen auf Twitter verglichen. Wenn es nun Abweichungen zwischen den Tweets auf Twitter und den in der Datenbank gibt, werden die Tweets in der Datenbank um die Abweichungen erweitert.

Im folgenden Abschnitt, wird der Algorithmus genau beschrieben und wie dieser in Form von Funktionen implementiert wurden ist.

3.2 Detaillierte Erklärung des Algorithmus zur Extration von Konversationen

In diesem Abschnitt wird detailliert erklärt, wie der Algorithmus aus einer Twitter Konversation einen Graphen in der Graph-Datenbank zeichnet.

Zu Beginn des Algorithmus wird die Funktion "startCollectingConversations()" aufgerufen. Welche wie der Name schon sagt, anfängt Konversationen zu sammeln. Zu Beginn der Funktion wird auf der Twitter API die Methode "getPlaceTrends()" aufgerufen. Diese Methode liefert von Twitter die aktuellen "Trends", je nach dem mit welcher "Location-woeid" die Methode aufgerufen wird. In diesem Fall wird die Methode mit der "Location-woid" = 23424829 aufgerufen, da die diese Ziffernfolge Deutschland steht. Somit erhält man schonmal die Twitter "Trends" aus Deutschland. Daraufhin wird zufällig eine Zahl zwischen 1 und 10 genommen. Diese Zahl bestimmt, mit welchem "Trend" aus Deutschland sich der Algorithmus in diesem Durchlauf beschäftigen soll. Wenn ein Trend dann ausgewählt wurden ist, wird geschaut ob dieser "Trend-" Knoten schon in der Datenbank liegt. Wenn dies der Fall ist, wird nichts gemacht und wenn dies nicht der Fall sein sollte, wird eine "Trend-" Knoten mit dem Namen des "Trends" angelegt. Dies hat den Vorteil, das gesammelte Konversationen ihren "Trends" bzw. Themen zugeordnet werden können. Aus diesem "Trend" Thema, werden dann über die Twitter API die ersten 100 Tweets geholt. Für diese ersten 100 Tweets, wird dann geschaut , welcher Tweet davon die meiste Interaktion hat. Wie schon gesagt ist das Messen der Interaktionen auf einen Tweet durch die Twitter API auf Retweeets und Linkes beschränkt. Jedoch ist es wahscheinlicher, wenn ein Tweet viele Retweets und Likes hat, das auch "Replies" stattegfunden haben. Denn somit scheint der Tweet interressant für andere User zu sein.

Wenn nun beliebter Tweet gefunden wurden ist, wird im nächsten Schritt geprüft, ob dieser Tweet schon in der Datenbank vorhanden ist. Dies muss gemacht werden, da bei jedem Durchlauf der "Trend" aus dem der Tweet kommt zufällig gewählt wurden ist. Somit könnte es ohne diese Prüfung zu Tweet Doppelungen kommen. Wenn dieser Tweet in der Datenbank gefunden wurden ist, wird mit dem nächsten Durchlauf gestarten und der Tweet somit verworfen. Wenn der Tweet nicht in der Datenbank gefunden wurden ist, wird fortgesetzt mit dem Abgreifen der Konversation rund um den Tweet.

Wenn der Tweet sich noch nicht in der Datenbank befindet, wird als nächstes geprüft ob der Tweet ein Retweet ist. Dies wird über die Twitter API mit der Methode "getRetweetedStatus()" geprüft. Wenn es kein Retweet ist, kann di-

rekt mit diesem Tweet weiter gemacht werden und ansonsten wird der "Original-" Tweet über die Twitter API geholt und damit weiter gemacht. Das holen des "Original-" Tweets ist über die "Query" durch die Methode(n) "getRetweetedStatus().getId()" möglich. Daraufhin wird der Tweet schonmal als Basis für die Konversation in die Graph-Datenbank abgelegt. An diesem Knoten wird die TweetId, der Text des Tweets, das Erstelldatum und eine Makierung ob es sich um den "Haupt-" Tweet der Konversation handelt gepeichert. Diese Felder sind vorallem wichtig, wenn es darum geht die Konversation zu erweitert. Darauf wird im verlaufe dieses Kapitels noch näher eingegangen.

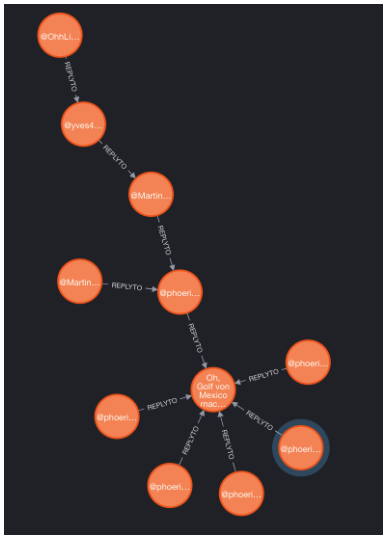
Beispiel eines Tweets in der Graph-Datenbank, welcher als Basis für eine Konversations rekonstruktions gilt. Mit den Feldern: tweetId: 1410731664506982408, title: The United States Supreme Court has undercut voting rights in America, createdAt:2021-07-01T22:46:53Z und rootTweet: true

Nach dem anlegen und abspeichern des Basis Tweets der Konversation, wird dieser Tweet auf mögliche Antworten untersucht. Somit wird sich die gesamte Konversation abbilden lassen. Da es wie in den vorherigen Kapiteln schon erwähnt, keine Methode in der Twitter API gibt um alle Anwtorten auf einen Tweet zu erhalten, müssen alle Tweets die mit dem Basis Tweet zusammenhängen einzeln geprüft werden.

Das holen aller Tweets die mit dem Basis Tweets in Verbindung stehen, kann gemacht werden in dem man mit der Twitter API nach der Basis TweetId sucht. Auf diesem Element, lässt sich dann eine Methode aufrufen, welche einem alle Tweets zu diesem Basis Tweet zur Verfügung stellt. "getAllTweets()" liefert eine Liste an Tweets. Jedoch ist damit nicht garantiert, dass das alles Antworten auf den Basis Tweet sind. Es könnten auch Antworten auf Antworten des Basis Tweets sein. Deswegen muss danch noch geprüft werden ob die Antworten in der Liste eine passende Antwort-Id zum Basis Tweet haben. Die kann über die Twitter API mit der Methode "getInReplyToStatusId()" erfolgen. Somit kann man dann die Antworten rausfiltern, welche wirklich Anwtorten auf den Basis Tweet sind.

Über diese Liste wird nun iterriert. Bei jeder Iterration, wird die Antwort gespeichert und als neuer Knoten in die Graph-Datenbank gespeichert. Dieser neue Knoten wird mit einer Verbindungslinie in Beziehung zu seinem Basis Tweet gebracht. Diese gespeicherte Antwort wird nun als neuer Basis Tweet betrachtet

um für diese Antwort auf den Basis Tweet, wiederrum Antworten zu finden. Um so einen Konversationsweg zu vervollständigen. Dieses Verfahren wird nun rekursiv in die Tiefe und iterativ über die Antwort Liste in die Breite ausgeführt um die gesamte Konversation abzubilden.



Beispiel einer mehrstufigen Konversation, rund um einen Basis Tweet. Der Basis Tweet ist mit 6 Antworten verbunden. Und auf einer der Anwtorten, gibt es wieder 2 Anwtorten. Wobei eine Antwirt nur ein Element tief ist und die andere Antwort 3 Elemente tief ist. Dies sind alles Antworten auf die vorherigen Antworten. Hierbei gilt es zu beachten, das jede Anwtwort ein eigenständiger Tweet ist, mit einer ReplyId auf seinen vorgänger Tweet.

Somit lassen sich Konversationen rund um Tweets aus Twitter abgreifen und als Graph darstellen. Im nächsten Abschnitt geht es darum, wie man Konversationen erweitern kann, nach dem diese schon in der Datenbank abgelegt wurden sind.

3.3 Detaillierte Erklärung des Algorithmus zur Vervollständigung von bereits extrahierten Konversationen

Nach dem eine Konversation von Twitter extrahiert wurden ist, wird die Methode "extendConversations()" aufgerufen, um alle in der Datenbank befindlichen Konversationen zu erweitern. Insofern es Erweiterungen in diesen Konversationen auf Twitter gibt.

Um die schon in der Datenbank vorhandenen Konversationen zu erweitern, werden zunächt erst einmal alle Knoten bzw. Basis Tweets aus der Datenbank geholt. Dies ist möglich, da beim anlegen der Knoten, wie oben erwähnt, gespeicht wurden ist ob es ein Basis Tweet ist oder eben nicht. Daraufhin wird diese Liste an Basis Tweets gefiltert nach dem Erstelldatum der Tweets. Denn Statistisch gesehen erhalten Tweets im laufe der nächsten 2 Stunden nach dem erstellen die meisten Anwtorten. Nach 6 Stunden befinden sich schon über 80 Prozent der Antworten in der Konversation. Und nach 12 Stunden befinden sich über 90 Prozent der Antworten auf der Konversation. Wie aus bereits vergangenen Arbeiten hervorgeht. Somit werden in dieser Arbeit alle Tweets erweitert, bei denen das Erstelldatum nicht länger als 12 Stunden her ist. Somit hat man die Möglichekti möglichst alle Antworten abzugreifen aber gleicheitzig nicht zu viele Konversationen zu haben, das der Algorithmus zum erweitern zu viel Zeit in Anspruch nehmen würde.

Durch die Filterung, können nun alle Tweets die nicht länger als 12 Stunden existieren, erweitert werden. Damit diese Tweets erweitert werden können, muss einzelnd über diese Tweets iterriert werden um diese zu analysieren. In jeder Iterration, wird zunächst geschaut, welche Antworten es auf Twitter zu diesem Tweet aktuell gibt. Diese Antworten werden dann mit den Knoten rund um den Basis Tweet verglichen und wenn es welche noch nicht in der Datenbank gibt, dann werden diese hinzugefügt. Es wird also für diese Antwort ein neuer Knoten hinzugefüt und eine Verbindungslinie zum Basis Tweets gezogen. Daraufhin wird dann das gleiche Schema auf alle Antworten des Basis Tweets angewendet. Hier werden dann alle Antworten des Basis Tweets durchgegangen und geschaut ob es zu diesem Tweet mehr Antworten gibt, als sich aktuell in der Datenbank befinden. Wenn ja werden diese wieder eingefügt. Dieser Schritt wird wieder rekursiv ausgeführt, um in der Tiefe an alle Antworten zu gelangen. So hat man ähnlich zu dem Verfahren in Abschnitt 3.2 einen Weg die Breite der Konversation, durch iteration, sowie die Tiefe der Konversation, durch rekursion zu erweitern.

Durch dieses Verfahren lassen sich Konversationen im nachhinein vervollständigen.

3.5 Verflechtung der Algorithmen aus Abschnitt 3.3 und Abschnitt 3.4

Um einen Datensatz an Konversationen sammeln zu können, wurden die Algorithmen aus Abschnitt 3.3 und Abschnitt 3.4 mit einander Verknüft. Es wird zu Beginn immer der Algorithmus aus Abschnitt 3.3 ausgeführt. Dieser legt dann eine Konversation von Twitter als Graph an. Daraufhin wird dann der Algorithmus von Abschnitt 3.4 ausgeführt, in dem dieser am Ende von Abschnitt 3.3 aufgerufen wurden ist. Nachdem der Algorithmus von Abschnitt 3.4 durchgelaufen ist, und somit alle in Konversationen die sich in der Datenbank befinden

erweitert hat, wird wieder der Algorithmus aus Abschnitt 3.3 aufgerufen. So ergibt sich ein immer weiter fortlaufendes Verfahren welchen nicht nur möglichst viele Konversationen sammelt, sondern diese auch immer wieder erweitert.

Die folgenden beiden Abschnitten in diesem Kapitel sind Erweiterungen, um in Kapitel 4 besser auf die Analysen des Datensatzen eingehen zu können.

3.6 Guppierung der Konversationen nach Themenzugehörigkeit

3.7 Grundlegenden Graph-Analysen