Der Twitterbot Vanellus zur vollständigen Analyse deutschsprachiger Tweets

von Chrilly Donninger

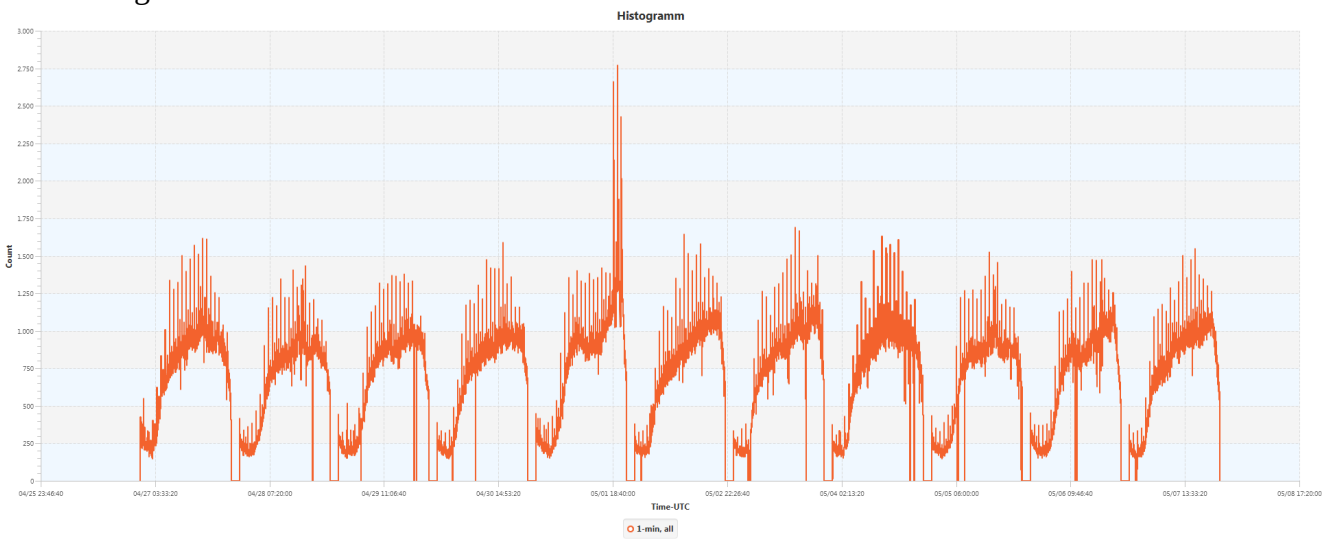
2018-05-08

Zusammenfassung:

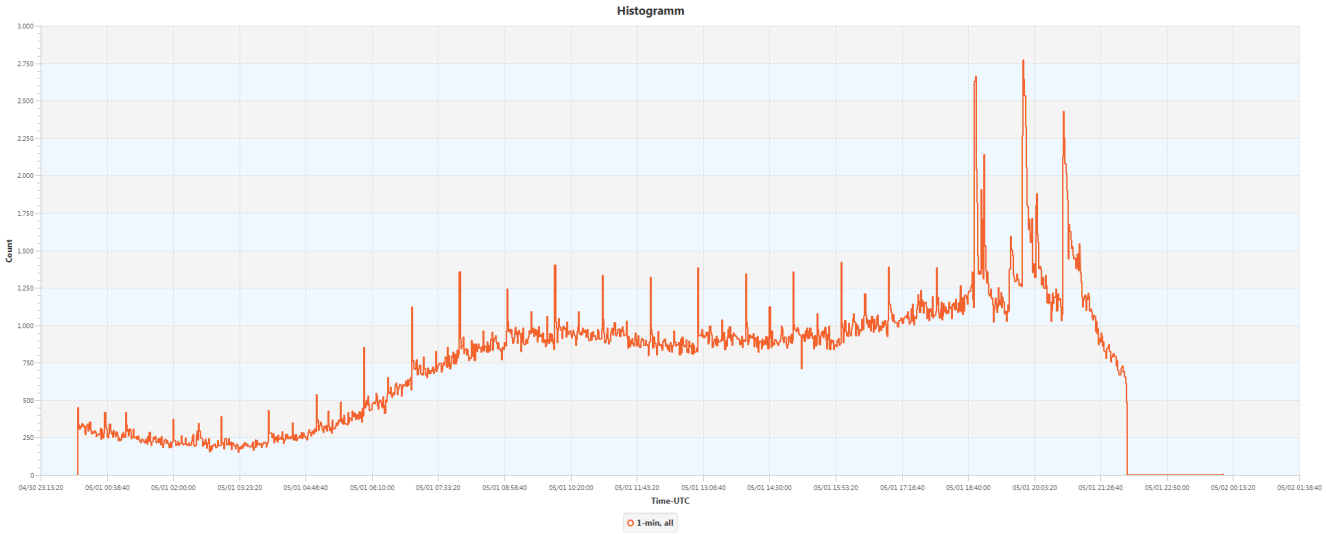
Vanellus ist der zoologische Namen für den Kiebitz. Es ist ein von Chrilly Donninger in JavaFX entwickelter Twitter-Bot. Dieses Dokument skizziert den Stand der aktuellen Version 0.29. Es werden mit dem Bot alle deutschsprachigen Tweets erfasst. Es wurden eine Reihe von Methoden zur Analyse dieser Daten entwickelt. Bis auf Test-Tweets greift Vanellus im Moment nicht aktiv in das Geschehen ein. Der Kiebitz hält – im Gegensatz zu seinem Schöpfer – bisher das Maul.
Das Dokument gliedert sich in 2 Teile. Im ersten werden die inhaltlichen Aspekte des Bots besprochen. Dieser Teil setzt nur eine (minimale) Vertrautheit mit Twitter voraus. Der zweiten Teil beschreibt die Architektur von Vanellus. Dieser Abschnitt/Anhang ist wahrscheinlich nur für mit der Informatik Vertraute von Interesse.

Die Daten:

Es wurden ab Freitag, 27. April, nach dem Twitter-Suchkriterium „*lang:de*“ alle Daten erfasst. Für diese Analyse werden die Daten bis einschließlich Montag, 7. Mai verwendet. An diesen 11 Tagen wurden in Summe 10.406.804 Tweets erfasst und zur weiteren Analyse gespeichert. Diese etwas mehr als 10 Millionen Tweets (nicht ganz 1 Million pro Tag) enthalten rund 7,5 Millionen unterschiedlichen Texte (ein Text wird im Mittel 1,4 Mal getweetet), die von 1,75 Millionen Usern (nicht ganz 6 Tweets/User) versendet wurden. Die Anzahl der Tweets pro User ist allerdings sehr schief verteilt. Die meisten versenden in diesen 11 Tagen nur 1-2 Tweets, es gibt jedoch ein paar „Power-User,“ die jeweils tausende Tweets erzeugen. Diese Power-User sind überwiegend Bots. Grafik 1 zeigt die Anzahl der Tweets pro Minute im Beobachtungszeitraum. Das tägliche Muster ist – bis auf den 1. Mai – sehr ähnlich. Es gibt zwischen den einzelnen Tagen immer einen Bereich, in dem keine Daten vorhanden sind. Der Twitter-Server ist – für Datenabfragen - zwischen 0:00 bis 2:00 Mitteleurop. Sommerzeit (CEST) außer Betrieb. Zumindest ist das für den von mir verwendeten freien Datenzugang der Fall. Allerdings fällt die Tweet-Rate nach einer Spitze zwischen 22:00 und 23:00 bis Mitternacht steil ab. Es ist in dieser Serverpause wohl nicht sehr viel los. Twitter verwendet diesen Zeitraum wahrscheinlich zur Wartung der Server.



Grafik 1: Anzahl der Tweets pro Minute vom 27. April bis zum 7. Mai 2018

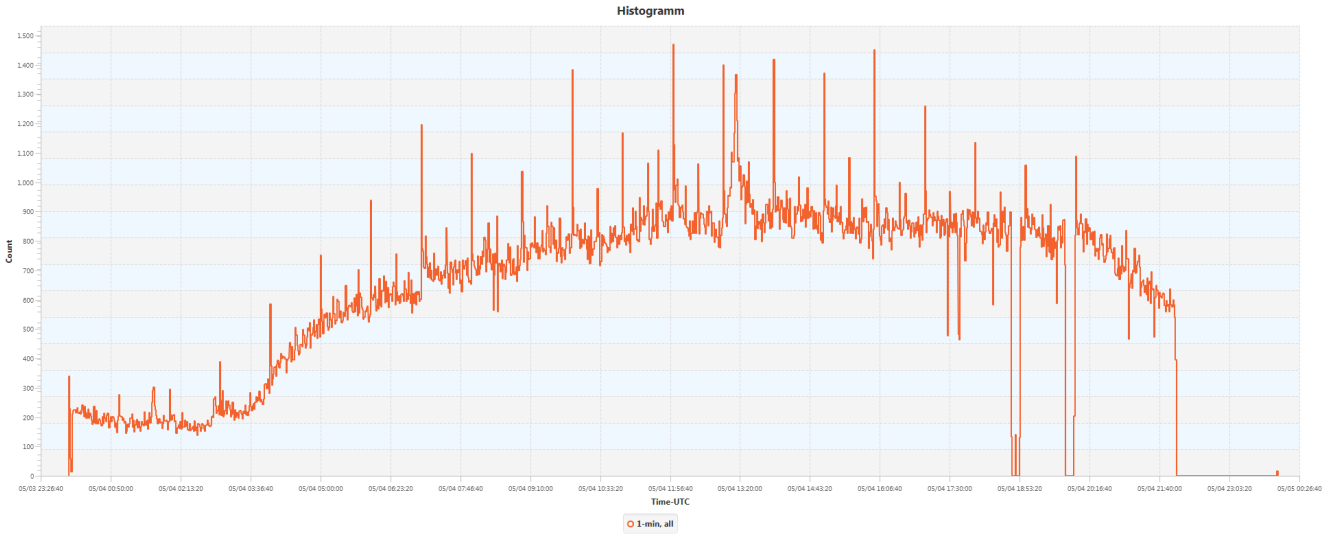


Grafik 2: Anzahl der Tweets pro Minute am Die, 1. Mai 2018

Grafik 2 zeigt die Tweets/min am Die, 1. Mai 2018. Von der typischen Rate von 1.000 Tweets/min weg gibt es am Abend klar erkennbare Spitzen bis 2.750 Tweets/min. Diese Spitzen entsprechen den Toren beim Champion-League Schlager Real-Madrid gegen Bayern-München. Man kann – indem man in Echtzeit nur die hereinkommenden Tweets zählt – unmittelbar auf wichtige Ereignisse schließen. Man weiß allerdings noch nicht, wer die Tore geschossen hat. Persönlich verwende ich diese Methode in meiner Eigenschaft als Hedge-Fund-Manager. Wenn der amerikanische Aktienindex S&P plötzlich fällt, schaue ich am Netz nach, was denn in der Welt passiert ist. Meistens tauchen die entsprechenden Meldungen erst etwas später auf. Die Aktienhändler haben die besseren News-Feeds.

Anmerkung: Vanellus verwendet allgemein den Twitter-Zeitstempel. Dieser ist in UCT (Greenwich). CEST ist UCT+2. Man muss zu den in der Grafik gezeigten Spitzen jeweils 2h dazu zählen. UCT hat u.A. den Vorteil, dass es keine Probleme bei der Umstellung von Sommer- zu Winterzeit gibt. Internationale Vergleiche sind nur in UCT sinnvoll.

Grafik 3 zeigt die Anzahl der Tweets/min am Freitag, 4. Mai. Man sieht am Abend zwei „*Aussetzer*“. Beim ersten meldete Twitter „*Server overloaded*“. Beim zweiten fiel meine Internet-Verbindung für ca. 5 Minuten aus. Diese kurzfristigen Internet-Aussetzer sind eine chronische Erscheinung meines ländlichen Internetzugangs. Sie sind bei meiner Trading-Software CashBot unter Umständen kritisch,



Grafik 3: Anzahl der Tweets pro Minute am Fr. 4. Mai Mai 2018

für eine statistische Analyse von Twitter-Daten jedoch nur etwas lästig. Twitter garantiert auch bei einem bezahlten Zugang nicht die vollständigen Daten. Diese Garantie wäre technisch kaum einlösbar. Es sind jedoch alle von Vanellus gesendete Testmeldungen wieder richtig angekommen.
Vanellus unterstützt alle Abfragen der Twitter-Schnittstelle. Man könnte auch in anderen Sprachen oder nach anderen Suchkriterien Daten herunter laden. Z.B. Nach Hashtags, Personen oder Parteien selektiert. Ich habe einigen Aufwand getrieben, um mit dem freien Zugang alle Deutschsprachigen Tweets erfassen zu können (siehe Anhang). Abgesehen von Twitter selbst dürfte es nicht sehr viele Datenbanken geben, die (fast) vollständig die deutschsprachige Twitterwelt erfassen. Es dürfte vor allem keine Datenbank geben, die das mit dem gratis-Zugang bewerkstelligt.

Datenabfrage:

Die Vanellus Datenbank unterstützt eine relativ mächtige Abfragesprache. Man kann sowohl nach Tweet-Texten, Usern und nach dem Tweet-Typ (Normaler Tweet oder Retweet) selektieren.
Auf die Abfrage „`ABBA AND Aldi`“ erhält man die folgenden Tweets (die Liste der Retweets ist noch wesentlich länger). „AND“ ist in der Abfrage eine logischer Operator. Es muss im Text sowohl das Wort „ABBA“ als auch „Aldi“ vorkommen. Unterstützt werden auch „OR“ und „NOT“. Bei NOT würden keine Texte ausgegeben, die das Wort „Aldi“ enthalten. Bei der Einstellung „Retweets none“ würde nur der originale und originelle Tweet des Users „*derLehnsherr*“ angezeigt werden.
(Die Zeitangaben sind in UCT).

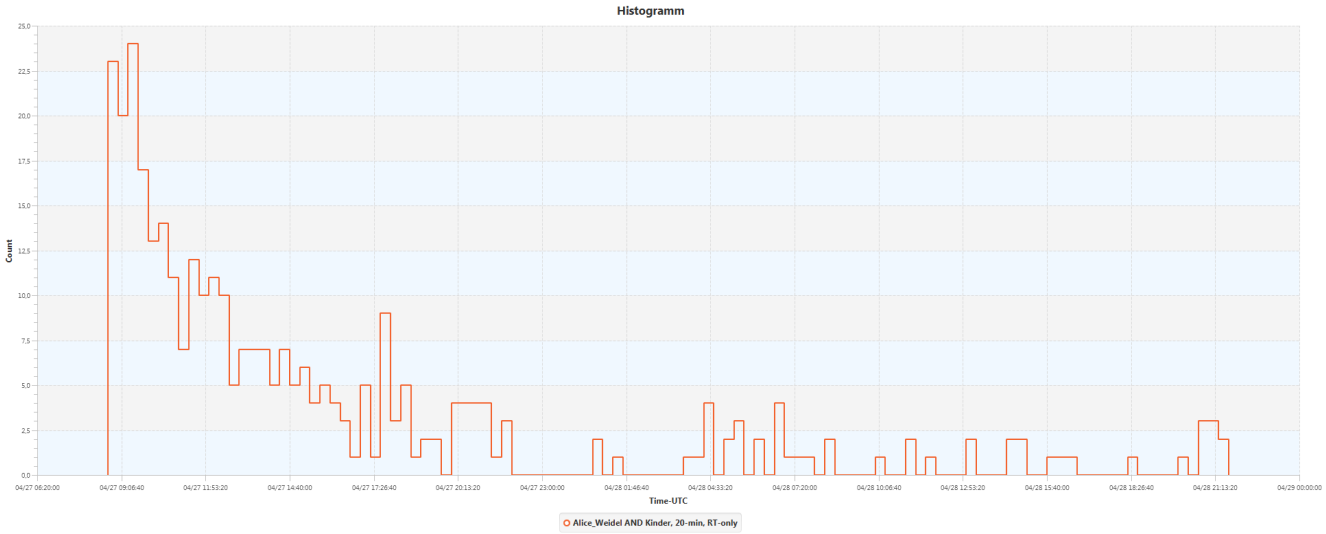
lang:de	04/27 13:07:17	derLehnsherr	ABBA: vereint Korea: fast vereint Jetzt wird es eng für Aldi Nord und Aldi Süd.
lang:de	04/27 13:07:40	PunkMetalOma	RT @derLehnsherr: ABBA: vereint Korea: fast vereint Jetzt wird es eng für Aldi Nord und Aldi Süd.
lang:de	04/27 13:07:39	jaimepenagos	RT @derLehnsherr: ABBA: vereint Korea: fast vereint Jetzt wird es eng für Aldi Nord und Aldi Süd.
lang:de	04/27 13:08:12	123StefB	RT @derLehnsherr: ABBA: vereint Korea: fast vereint Jetzt wird es eng für Aldi Nord und Aldi Süd.
lang:de	04/27 13:09:21	stefanie_579	RT @derLehnsherr: ABBA: vereint Korea: fast vereint Jetzt wird es eng für Aldi Nord und Aldi Süd.
lang:de	04/27 13:09:47	boundborg	RT @derLehnsherr: ABBA: vereint Korea: fast vereint Jetzt wird es eng für Aldi Nord und Aldi Süd.

An dem Tag als die Twitter-Gemeinde die Wiedervereinigung von ABBA bewegt hat, haben die beiden AfD Politikerinnen Alice Weidel und Beatrix v. Storch das untenstehende getweeted bzw. retweeted.

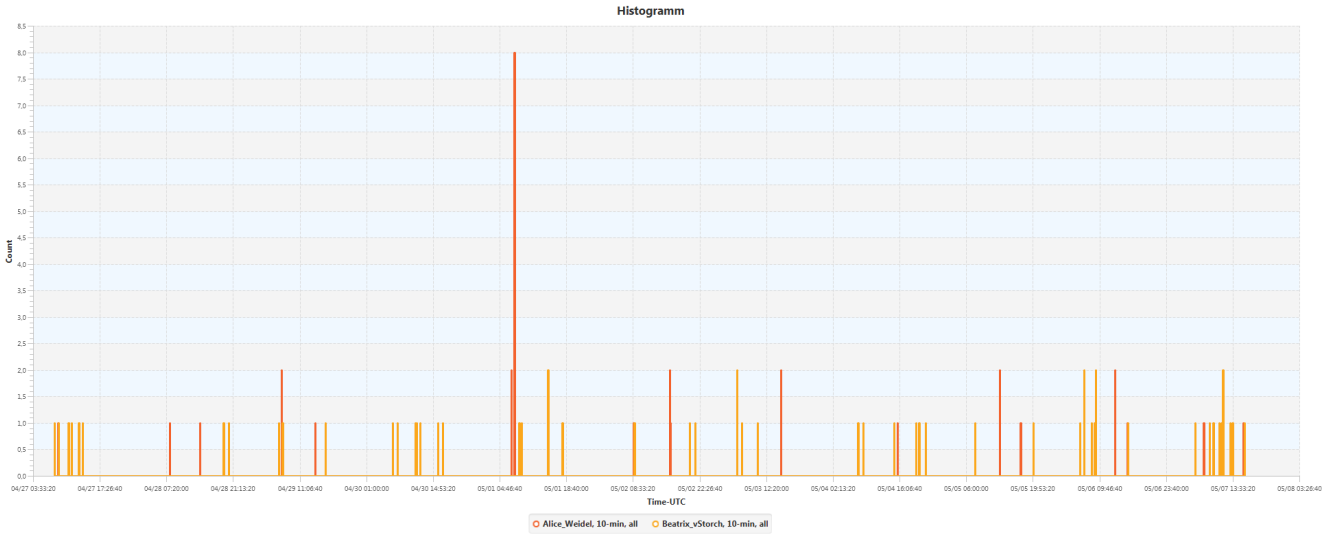
lang:de	04/27 08:04:55	Beatrix_vStorch	„Statt eines Kruzifixes sollten wir in öffentliche Gebäude lieber eine Fritz-Box hängen“ #FDP @GydeJ https://t.co/FkYxKxZzEm
lang:de	04/27 08:42:17	Alice_Weidel	Kinder sind unsere Zukunft & Familien das Rückgrat einer jeden Gesellschaft! Ministerpräsident Orbán betreibt seit Jahren zukunftsorientierte Familienpolitik, während die Bundesregierung die demographische Zuspitzung ...
lang:de	04/27 08:56:06	Beatrix_vStorch	Im #Bundestag traut sich die #FDP Kippa zu tragen- wie heroisch. Geht doch damit mal über den Hermannplatz in#Berlin #Neukölln! https://t.co/9O1ucMWbjq
lang:de	04/27 10:59:23	Beatrix_vStorch	RT @AfDBerlin: Muster-Notunterkunft steht leer, Kosten laufen bis 2025! Sinnbild für eine planlos wirkende Strategie der Integrationssenato...
lang:de	04/27 11:08:01	Beatrix_vStorch	RT @AfDimBundestag: 1. #Bundestagsrede der #AfD-MdB @VerHartmannAfD zur "Lage der Religions- und Weltanschauungsfreiheit": In der 1. Deba...
lang:de	04/27 11:32:06	Beatrix_vStorch	RT @AfDimBundestag: Stellv. #AfD-Fraktionsvorsitzende @Beatrix_vStorch zu "70 Jahre Gründung des Staates I #Israel": Seitdem die Bundeska...
lang:de	04/27 13:03:37	Beatrix_vStorch	RT @UdoHemmelgarn: Migranten-Angriffe auf Rettungskräfte! Sogar die 'tz' gibt zu: "Oft handelt es sich dabei um #Einwandererkinder der...
lang:de	04/27 13:14:53	Beatrix_vStorch	RT @PetrBystronAfD: Keine #Islamisierung, nirgendwo? Ein kurzes Zitat reicht und das Lügengebäude stürzt ein. #MutzurWahrheit #Christenverf...
lang:de	04/27 13:57:12	Beatrix_vStorch	RT @Dieter_Stein: Eine starke Rede von @Beatrix_vStorch . Ich bin gespannt, ob die Bundesregierung die Zahlungen an die @UNRWA aufrecht erh...

Alice Weidel erzielte mit ihrem „*Die Kinder sind unsere Zukunft*“ Tweet 376 Retweets. Wobei ein paar User wiederholt die Retweet Taste gedrückt haben. Grafik 4 zeigt das zeitliche Verhalten der Retweets. Wie üblich erfolgen die Reaktionen unmittelbar. Sie fallen im Laufe des Tages (steil) ab. In diesem Fall gab es auch am folgenden Tag noch Retweets. Das ist eher ungewöhnlich und hängt vermutlich mit dem Wochenende zusammen. In diesem Fall gab es vereinzelt noch Retweets in der folgenden Woche. Aus Gründen der besseren Darstellung wird in Grafik 4 nur Freitag und Samstag betrachtet.
Grafik 5 zeigt das zeitliche Twitter-Verhalten von Alice Weidel (rot) und Beatrix v. Storch (gelb). v. Storch twittert regelmäßiger. Das dürfte auch daran liegen, dass sie mehr Retweeted während Alice Weidel Retweets eher vermeidet. Alice Weidel war nur am Tag der Arbeit ganz zeitig in der Früh (um 6:30 CEST) in erheblicher Twitterstimmung. An normalen Arbeitstagen twittert sie eher um 10Uhr (CEST).

Anmerkung: Man könnte derartige Analysen für jeden beliebigen Politiker bzw. User machen. Ich habe die beiden Damen gewählt, weil sie in Deutschland markante Social-Media-Figuren sind.



Grafik 4: Anzahl der Retweets pro 20 Minuten auf den „*Kinder-Tweet*“ von Alice Weidel



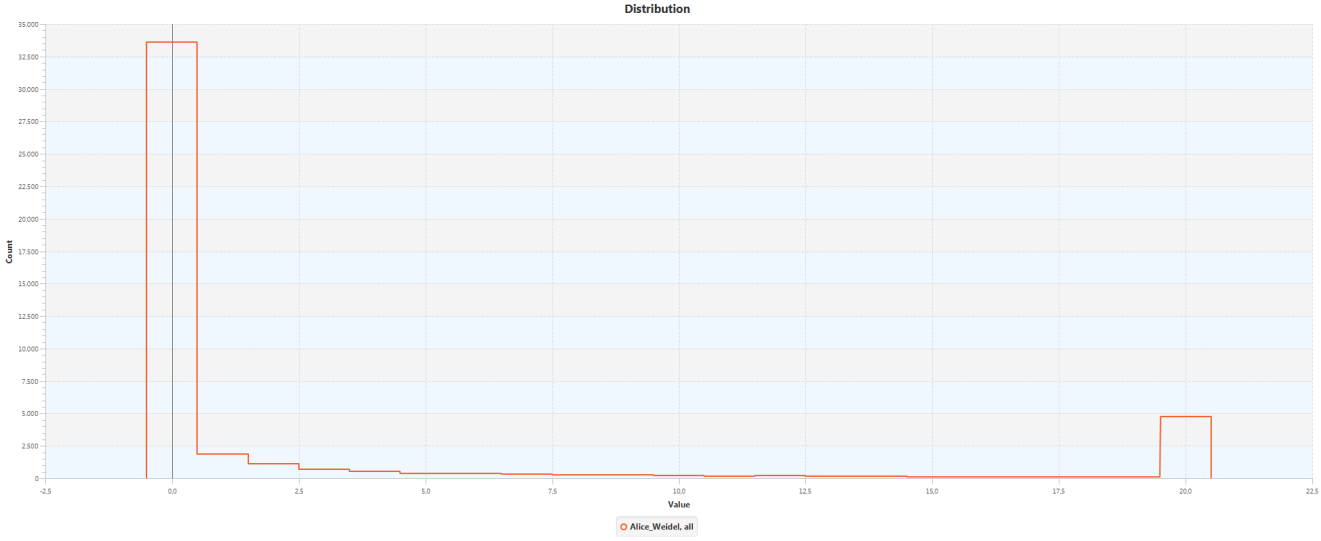
Grafik 5: Histogramm der Tweets von Alice Weidel (rot) und Beatrix v. Storch (gelb)

Followers:

Alice Weidel hatte beim Abfassen dieses Beitrages (Die. 8. Mai) 45.695 Followers. Bei Beatrix v.Storch waren es 36.612. Grafik-6 zeigt die Anzahl der User die in den 11 Tagen 0, 1, 2...20 und mehr Tweets verschickt haben. Die mit Abstand größte Gruppe mit rund 33.500 Usern sind die gänzlich Inaktiven. Man könnte jemanden, der innerhalb von 11 Tage keinen einzigen (Re-)tweet – egal zu welchen Thema - absetzt auch als „*Karteileiche*“ bezeichnen. Das heißt nicht, dass er die Tweets von Alice Weidel auf alle Fälle ignoriert. Sein Interesse für Twitter dürfte jedoch begrenzt sein. Auf Facebook gibt es die Möglichkeit, Follower zuzukaufen. Ich weiß nicht, ob das auch auf Twitter der Fall ist.

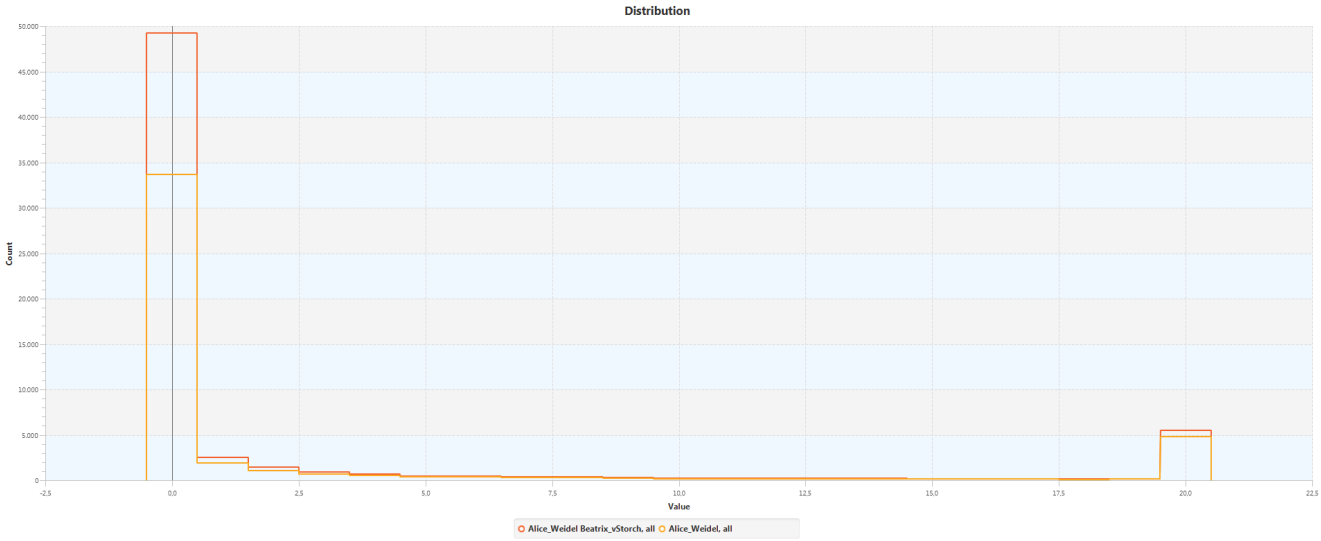
Rund 2.000 Follower haben 1 Tweet abgesetzt. Die Anzahl fällt mit steigender Tweets-Zahl kontinuierlich ab. Die größte Gruppe mit knapp 5.000 Mitgliedern bilden die Power-User mit 20 und mehr (Re-)tweets. Man könnte in einer weiteren Stufe das Verhalten dieser User betrachten und so auf den Gesamt-Verbreitungsgrad kommen. Dies wurde für diese Analyse noch nicht gemacht und ist auch in Vanellus noch nicht implementiert. Power-User sind aber oftmals Bots die gnadenlos Retweeten was ihnen unter die Finger kommt (siehe das Bots-Kapitel weiter unten). Es gibt unter den Power-Usern

jedoch auch AfD Fans und Gegner die ein erhebliches Sendungsbewusstsein haben. Die Verteilung von Beatrix v. Storch ist – auf entsprechend niedrigeren Niveau – so gut wie identisch mit jenem ihrer Parteifreundin.



Grafik 6: Anzahl Tweets der Alice Weidel Followers vom 27. April bis 7. Mai.

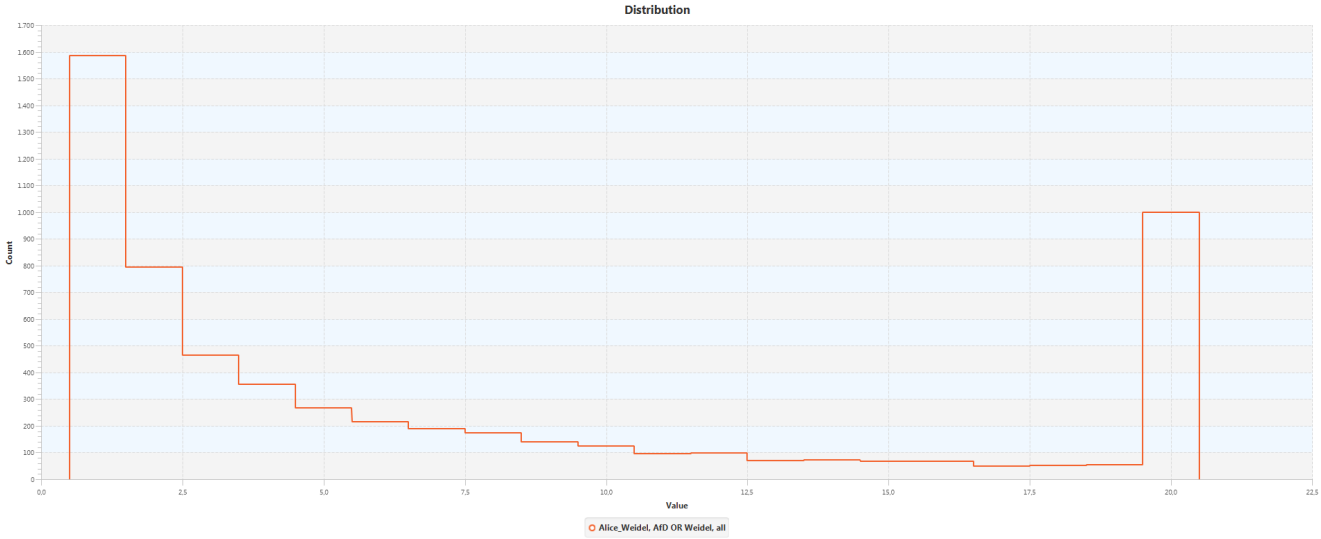
Fasst man die Follower von Weidel und Storch zusammen (rot), dann erhöht sich gegenüber den Followern von Alice Weidel (gelb) die Anzahl der aktiven User nur unwesentlich. Es überschneidet sich natürlich auch ein Teil der Karteileichen.



Grafik 7: Anzahl Tweets der Alice Weidel und Beatrix v. Storch Followers vom 27. April bis 7. Mai.

Bisher wurde alle Tweets betrachtet. Es könnte sich auch um „Tooor“ bei Real gegen Bayern gehandelt haben. Grafik 8 zeigt die Anzahl der aktiven A. Weidel User, in deren Tweet-Text entweder das Wort „Weidel“ oder „AfD“ vorkommt. Die Anzahl der politisch aktiven Usern ist natürlich noch etwas geringer.
Vanellus sucht genau genommen nach Substrings. Für den Suchbegriff „AfD“ gilt der Hashtag „#AfD“ oder „#AfDwirkt“ ebenfalls als Treffer. Im Falle von B.v.Storch könnte es sich auch um den Vogel handeln. Allerdings gibt es – von den politischen Gegnern – auch Wortspiele mit der Doppelbedeutung. Z.B. „*Niemand möchte ein Baby von Storch*“. Eine Erkennung ob der Vogel oder die Politikerin gemeint ist, ist für einen Bot eine extrem schwierige bis unlösbare Aufgabe. Die Unterscheidung ist

akademisch sehr reizvoll, die statistische Analyse der Tweets würde sich jedoch kaum verändern. „*Storch*“ ist für Vanellus immer Beatrix von. Grafik 8 zeigt nur die aktiven Benutzer an. Wenn man links noch den Balken der (politisch) Inaktiven zeichnet, sieht man rechts so gut wie nichts mehr. Bemerkenswert ist vor allem, dass bei der Selektion von politisch aktiven Usern vor allem die Anzahl der „*Power-User*“ sehr stark von 5.000 auf 1.000 sinkt.



Grafik 8: Anzahl Tweets der Alice Weidel mit „*AfD oder Weidel*“ im Text.

Bot-Erkennung:

Speziell seit dem US-Präsidentenwahlkampf 2016 gibt es eine intensive Diskussion um den Einfluss von Social-Media-Bots. Es hat auch das amerikanische Verteidigungsministerium einen Wettbewerb zur Bot-Erkennung ausgeschrieben. Die Diskussion um diese Frage hat mein Interesses für dieses Thema geweckt. Als alter Hacker bin ich immer auf der Suche nach interessanten Rätseln und Herausforderungen. Der nicht ganz Ernst gemeinte Projekttitel ist „*Highland Analytics*“. (Ich wohne in der Region Waldviertel-Hochland, es gibt u.A. den Reiter-Treff „Highland Ranch“).
Es gibt verschiedene Ansätze zur Bot-Erkennung. Der klassische Versuch ist inhaltlicher Natur. Man versucht durch Spracherkennung und Sprachanalyse den Bots auf die Spur zu kommen. Der andere ist rein Statistisch. Man untersucht das Zeitverhalten und auch die Tatsache, dass Bots in der Regel im Rudel auftreten. Man kann beide Ansätze verbinden. Soweit ich das beurteilen kann, hat sich der statistische Ansatz als der effektivere erwiesen. Vanellus verfolgt – zumindest bisher – die rein statistische Methode.

1) Text-Zwillinge:

Unter einem Text-Zwilling versteht Vanellus zwei User die ungefähr die selbe Anzahl von Tweets senden und die eine hohe Anzahl von Tweets gemeinsam haben. Für diese Untersuchungen darf die Anzahl der Tweets um maximal 5% abweichen und sie müssen mindestens 150 gemeinsame Tweets haben.
Anmerkung: Selbst bei zwei vollkommen identischen und synchron laufenden Bots würde man Abweichungen messen, da die Bots Probleme beim Abschicken des Tweets haben können, der Twitter-Server Tweets verschluckt und nicht zuletzt meine Internetverbindung nicht 100% stabil ist.

Auto-Zwillinge:

Zu meiner Überraschung erzeugte die Methode zur Erkennung von Zwillingen auch sogenannte Auto-Zwillinge. Ein Bot ist der Zwilling von sich selbst. Das passiert, wenn ein Bot dieselbe Nachricht mehrmals schickt. Ein Beispiel dafür ist „*phil_de_bot*“. Dieser Bot verschickt alle 30 Minuten einen

Tweet mit einem zufällig ausgewählten philosophischen Zitat. Der Zitatenschatz ist nicht sehr groß und so wiederholt „*phil_de_bot*“ regelmäßig Zitate. Es gibt Bots, die Kirchturm-Glocke spielen und jede volle Stunde die Zeit durchgeben. Auch diese Meldungen wiederholen sich periodisch. Ein anderes Beispiel sind Radiosender, die den jeweiligen Liedtitel tweeten. Aktuelle Hits werden regelmäßig gespielt und es wiederholt sich dementsprechend der Tweet. Die Auto-Zwillinge sind mir passiert. Ich haben den Bug (Fehler) zum Feature gemacht. Man kann die Erkennung auch ausschalten bzw. nur nach Auto-Zwillingen suchen. Ich fand die Erkennung der Auto-Zwillinge am Ende durchaus interessant und habe dieses ungeplante Feature daher im Kode gelassen.
Es gibt Bots die sowohl Auto- als auch echte Zwillinge sind. Sie wiederholen eine Botschaft mehrmals und tun dies zusätzlich im Rudel.

Echte Zwillinge:

Bei echten Zwillingen handelt es sich um zwei verschiedene User. Wobei diese in der Regel ganze Rudel bilden. Ein Beispiel dafür ist der Homöopathie-Rudel.
Der User „*sand_iris*“ bildet einen Zwilling mit „*Ann_Walters_*“, „*Barbara_Hofer_*“, „*H_Schneider_*“ …. Diese bilden mit weiteren Usern wieder Zwillinge. Der Screenshot zeigt die gesamte Homöopathie-Gang. Die Gang ist nach dem von Vanellus zuerst gefunden Mitglied benannt. Ziel dieser Gang ist die Propagierung der Homöopathie. Der Screenshot unterhalb zeigt die häufigsten Hashtags von „*sand_iris*“. Die zweite Spalte in der Tabelle enthält die Anzahl der Tweets von User-1, die vierte Spalte jener von User-2, die 5-te Spalte die Anzahl der gemeinsamen Tweets und die letzte Spalte den „*Gang-Namen*“.
Es hat eine gewisse Ironie, wenn Anhänger der sanften Medizin ein umfangreiches Bot-Netzwerk zur Propagierung ihrer Ansichten aufbauen.

sand_iris	205	Ann_Walters_	207	182	sand_iris
sand_iris	205	Barbara_Hofer_	203	173	sand_iris
sand_iris	205	H_Schneider_	205	170	sand_iris
sand_iris	205	Clara_Becker_	198	175	sand_iris
sand_iris	205	Mary_Hahn_	201	173	sand_iris
Werner_Schulz_	192	Barbara_Hofer_	203	164	sand_iris
Werner_Schulz_	192	Clara_Becker_	198	175	sand_iris
Werner_Schulz_	192	Mary_Hahn_	201	167	sand_iris
Barbara_Hofer_	203	H_Schneider_	205	166	sand_iris
Barbara_Hofer_	203	Clara_Becker_	198	169	sand_iris
Barbara_Hofer_	203	Mary_Hahn_	201	174	sand_iris
Clara_Becker_	198	H_Schneider_	205	171	sand_iris
Ann_Walters_	207	Barbara_Hofer_	203	177	sand_iris
Ann_Walters_	207	H_Schneider_	205	171	sand_iris
Ann_Walters_	207	Clara_Becker_	198	175	sand_iris
Ann_Walters_	207	Mary_Hahn_	201	178	sand_iris
Mary_Hahn_	201	Clara_Becker_	198	173	sand_iris
Mary_Hahn_	201	H_Schneider_	205	166	sand_iris

148	Hashtags	#Homöopathie
12	Hashtags	#MachAuchDuMit
10	Hashtags	#machauchdumit
8	Hashtags	#homeopathy
7	Hashtags	#Homeopathy
6	Hashtags	#Globuli
4	Hashtags	#Placebo

Die häufigsten Hashtags von User „*sand_iris*“

Die mit Abstand größte Bot-Gang ist „*Arikatytm31*“. Diese Gang hat einige Dutzend hoch aktive Mitglieder. Der Screenshot zeigt die ersten 10 Tweets dieses Users. Alle Tweets dieser Gang enthalten nur Hashtags. Offensichtliches Ziel ist das Hashtag-“*pushen*“. Unter den Top-10 der Hashtags hat die Gang die Position 1, 5, 6 und 7 inne. Der Spitzenreiter „*#PremiosMTVMiaw*“ hat den von „*#AfD*“ angeführten Pulk weit abgehängt. Ich kenne mich in der Welt der Social Media zu wenig aus, um beurteilen zu können, welchen Zweck es hat, mit „*#PremiosMTVMiaw*“ die Poleposition zu erringen. Angeblich nimmt Twitter sozial besonders auffällige Bots bzw. Bot-Gangs aus dem Rennen. Bei dieser sehr leicht erkennbaren Gang ist dies nicht der Fall.

lang:de	04/27 00:13:32	Arikatytm31	RT RTLisaFitzpatrick #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 00:28:22	Arikatytm31	RT Scottdrives #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 00:46:04	Arikatytm31	RT KristinePThomas #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 00:46:09	Arikatytm31	RT glmh101 #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 00:46:12	Arikatytm31	RT mindwanderer #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 13:38:22	Arikatytm31	RT RTgbrodermann #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 13:5[illegible]23	Arikatytm31	RT sofiaberen #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 14:08:24	Arikatytm31	RT RT DizzyDortch #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 14:08:30	Arikatytm31	RT RTDizzyDortch #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC
lang:de	04/27 14:23:11	Arikatytm31	RT RT JasonMehrtens #MTVBRPETNUGGET #MTVBRSHADETAYLORKATY #PremiosMTVMiaw #MTVLAINSTAGLCAMILAC

63364	Hashtags	#PremiosMTVMiaw
42399	Hashtags	#AfD
42216	Hashtags	#rp18
36827	Hashtags	#ReconquistaInternet
33999	Hashtags	#MTVLAINSTAGLCAMILAC
33849	Hashtags	#MTVBRPETNUGGET
33842	Hashtags	#MTVBRSHADETAYLORKATY
30917	Hashtags	#RMAFCB
29088	Hashtags	#Ellwangen
26581	Hashtags	#

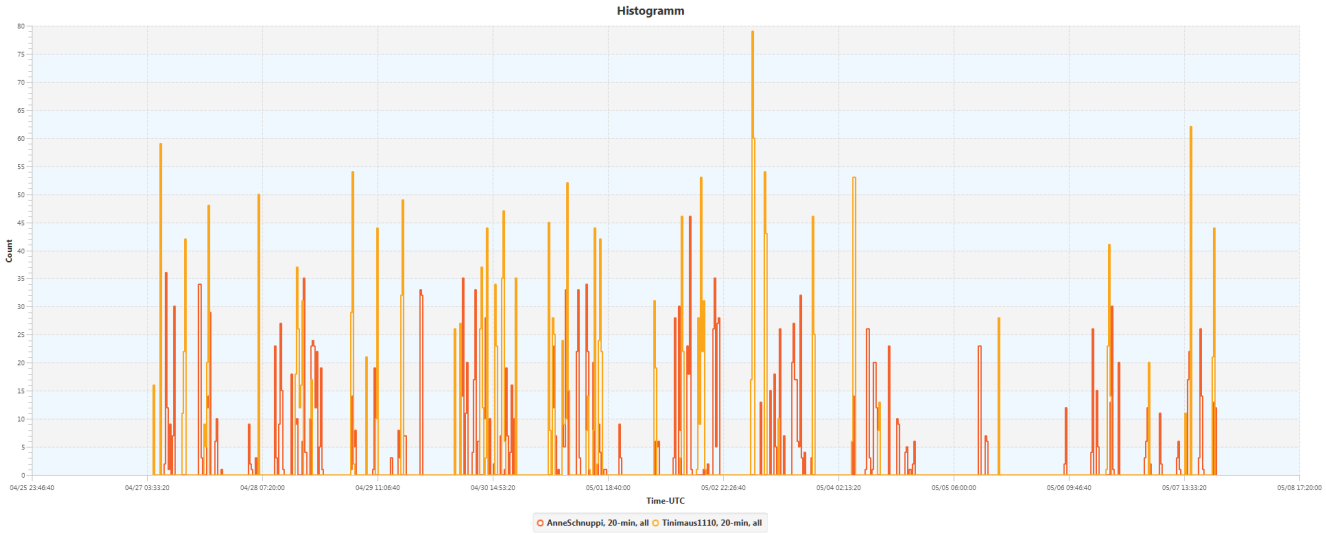
Eine politisch motivierte Gruppe bildet „*AnneSchnuppi*“. Siehe den Screenshot auf der folgenden Seite. Hier ist es jedoch unklar, ob es sich um eine Gruppe von Bots oder um besonders fleißige Menschen handelt. Die Gruppe twittert pro User am Tag etwas mehr als 200 Tweets. Die Übereinstimmung ist jedoch im Verhältnis zur Anzahl der Tweets relativ niedrig. Typischer Weise sind dies dieselben Retweets. 200 Retweets/Tag sind für einen Menschen zu schaffen. Grafik 9 zeigt das zeitliche

Verhalten von *AnneSchnuppi* und *Tinimaus1110*. Es gibt gemeinsame Perioden hoher Aktivität. Tinimaus1110 schafft bis zu 80 Tweets in 20 Minuten. Innerhalb 1 Minute bringt es *Tinimaus1110* auf bis zu 9 Tweets. Das ist für einen Menschen nicht gänzlich unmöglich. Vermutlich sind die Mitglieder der AnneSchnuppi-Gang Profis, die eine gewisse Anzahl von Tweets abliefern müssen. Die Mitglieder hauen kurz in die Tasten rein, haben dann wieder 1-2 Stunden Pause, bevor sich das Spiel wiederholt. Sie könnten aber auch einer anderen Tätigkeit – eventuell ist es sogar richtige Arbeit – nachgehen und in den Pausen eine AfD-Session einlegen. Die dritte Möglichkeit ist eine Bot-Familie dessen Zeit- und Synchronisationsverhalten etwas intelligenter ist als bei den meisten übrigen Bots. Wobei etwa ein Zeitungs- oder Radiobot gar nicht verschweigen will, dass er ein solcher ist.

Tinimaus1110	2424	LC180666	2446	204	AnneSchnuppi
AnneSchnuppi	2338	artepmobil	2458	213	AnneSchnuppi
AnneSchnuppi	2338	GertWalterWolf1	2258	213	AnneSchnuppi
AnneSchnuppi	2338	Tinimaus1110	2424	262	AnneSchnuppi
AnneSchnuppi	2338	IsidorMeyer1	2418	165	AnneSchnuppi
IsidorMeyer1	2418	Tinimaus1110	2424	308	AnneSchnuppi
EEffektiv	2295	GertWalterWolf1	2258	168	AnneSchnuppi
artepmobil	2458	IsidorMeyer1	2418	232	AnneSchnuppi
artepmobil	2458	LC180666	2446	186	AnneSchnuppi
artepmobil	2458	Tinimaus1110	2424	325	AnneSchnuppi

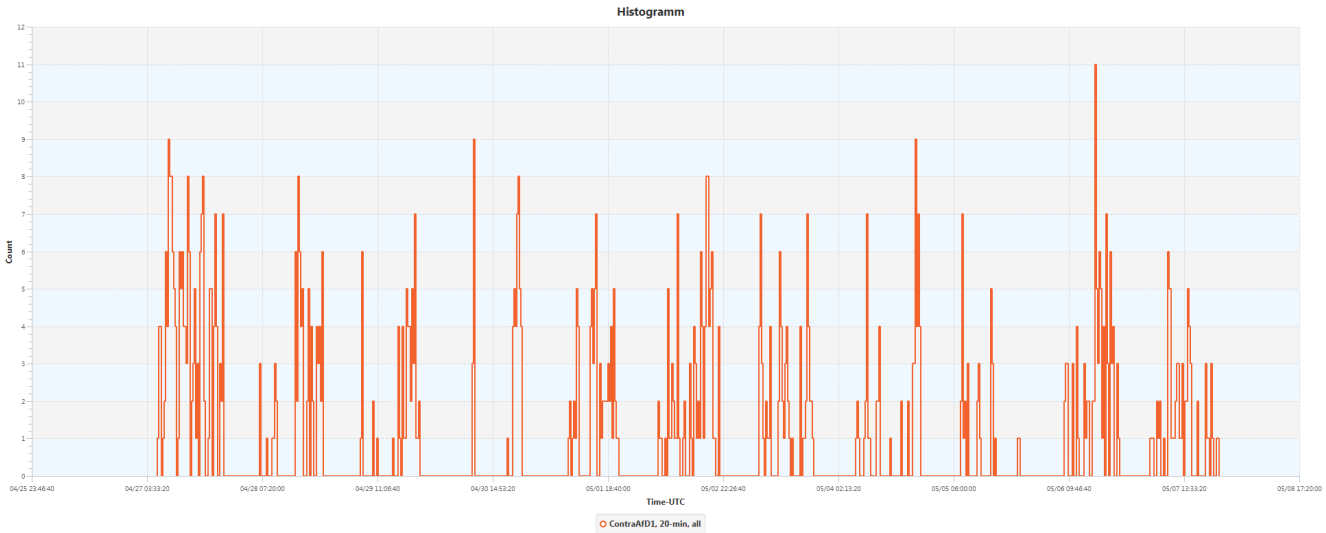
29	Hashtags	#Ellwangen
28	Hashtags	#AfD
20	Hashtags	#Merkel
14	Hashtags	#Polizei
11	Hashtags	#Berlin
11	Hashtags	#ZDF
11	Hashtags	#Syrien
11	Hashtags	#Deutschland

Die häufigsten Hashtags von „*AnneSchnuppi*“.



Grafik 9: Histogramm von *AnneSchnuppi* (rot) und *Tinimaus1110* (gelb) mit 20 min. Zeitintervall.

Die meisten Power-User zum Thema AfD sind Fans. Es gibt jedoch mit „*ContraAfD1*“ einen sehr aktiven Gegner. *ContraAfD1* tweeted längere Perioden durch, bevor er wieder eine Pause einlegt. Seine Spitzenleistung ist mit 11 Tweets in 20 Minuten auch wesentlich niedriger als etwa von *Tinimaus1110*. *ContraAfD1* retweeted kritische AfD Kommentare, verfasst aber auch relativ viel selber. Das dauert natürlich länger als auf den Retweet Knopf zu drücken. Er hat ebenfalls sehr viel Tagesfreizeit, beginnt sein Tagwerk früh und hört am Abend relativ früh auf. Die große Masse der User legt hingegen kurz vor dem Schlafen gehen, zwischen 22 und 23Uhr CEST, noch eine Twitterrunde ein. Es dürfte sich um einen alten Linken handeln, der sich in der Pension zur Aufgabe gemacht hat, die AfD zu bekämpfen. Seine Kommentare sind in der Regel etwas subtiler als jene der AfD Fans. Wobei aber – im Sinne von Marshall McLuhans „*The medium is the message*“ - das Medium selbst keine besonders ausgefeilten Kommentare zulässt. Das dürfte auch die Beliebtheit von Twitter in der politischen Kommunikation erklären.



Grafik 10: Histogramm von *ContraAfD1* mit 20 min. Zeitintervall.

Vanellus kann von sich aus noch nicht zwischen AfD-Fans und einen Gegner wie *ContraAfD1* unterscheiden. Diese – für einen Menschen triviale – Einschätzung stammt von mir. Eine automatische Klassifikation ist jedoch geplant. Ich habe aus diesem Grund auch das Thema AfD gewählt, weil es fast immer ein klares Pro- und Contra gibt und sich die Erkennung nicht mit einer Grauzone herumschlagen

muss. Es ist für den Entwickler leicht zu entscheiden, ob der Bot mit seiner Klassifikation richtig liegt. Die Hashtags von *ContraAfD1* geben bereits einen gewissen Hinweis, aber es dominiert auch hier – so wie bei den Fans - der Hashtag *#AfD*. Ein wichtiger Anhaltspunkt sind Retweets. Ein Gegner wird wohl kaum Alice Weidel oder Beatrix v. Storch retweeten. Allerdings retweeted *ContraAfD1* kritische Tweets die ihrerseits wieder Teile des Original-Tweets verwenden. Die Unterscheidung ist für einen Menschen einfach, für einen Bot jedoch eine sehr harte Nuss. Die Klassifizierung ist unter der Bezeichnung „*Sentiment Analysis*“ ein sehr aktives Feld der Forschung. Schwerpunkt der „*Sentiment Analysis*“ sind Produkt-Besprechungen. Wobei die für die Industrie interessanten Kommentare in die Grauzone fallen. Z.B. „*Die Kamera ist preiswert, leicht und macht schöne Bilder, der Akku könnte jedoch länger halten*“. Die Erkennung von Bot-Spam ist auch auf diesem Gebiet sehr wichtig.

88	Hashtags	#AfD
17	Hashtags	#noAfD
16	Hashtags	#gera
8	Hashtags	#storch
7	Hashtags	#reil
7	Hashtags	#noNazis
6	Hashtags	#noafd
6	Hashtags	#Gera
5	Hashtags	#c0105
4	Hashtags	#laudenbach
3	Hashtags	#Querfurt
3	Hashtags	#ReconquistaInternet

Die häufigsten Hashtags von „*ContraAfD1*“.

Vanellus hat noch zahlreiche andere Gangs identifiziert. Z.B. tweetet eine Bot-Gang periodisch die User-Namen der anderen Mitglieder. Offensichtlich dient das zur Synchronisation der Mitglieder. Die Gang zeigt jedoch ansonsten keinerlei Twitter-Aktivität. Möglicher Weise nutzt sie die Synchronisation um auf anderen sozialen Kanälen koordiniert aktiv zu werden. Es gibt u.A. auch eine zu *AnneSchnuppi* analoge Gang von Erdogan Fans. Das zeitliche Verhalten ist sehr ähnlich. Man könnte – analog zur Untersuchung der Followers von Alice Weidel – die Wirkung derartiger Gangs weiter verfolgen. Im Moment muss man das händisch machen. Es war mir zu mühsam. Man könnte es jedoch relativ leicht als automatisches Feature implementieren.

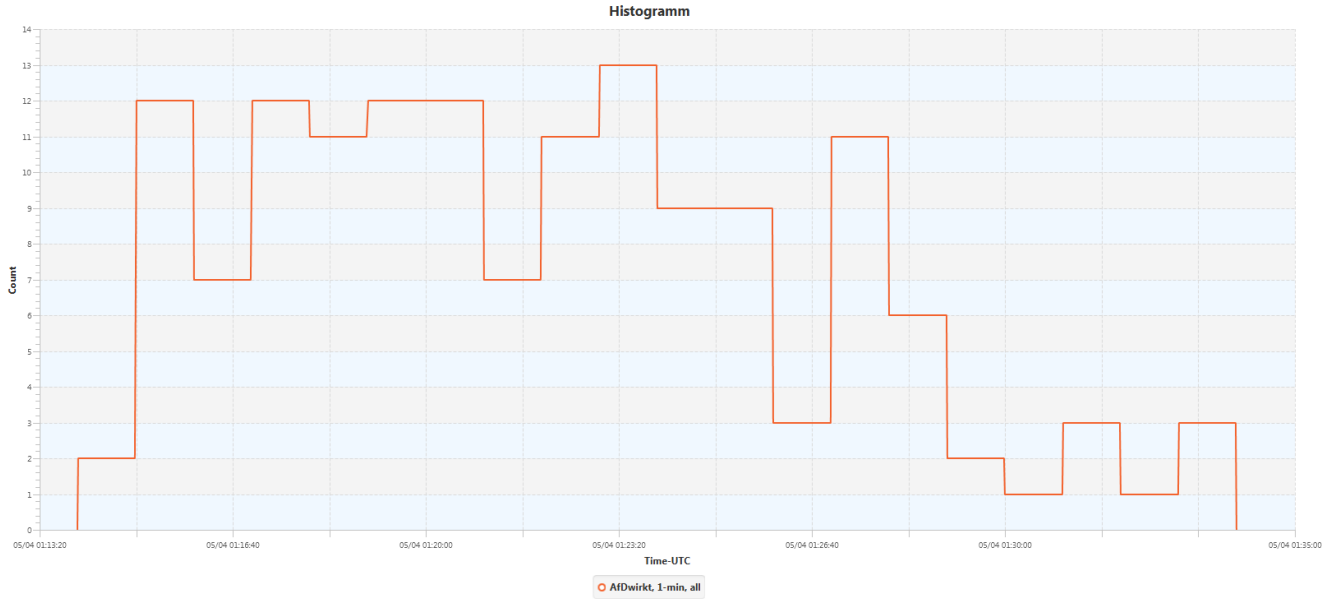
Die Twitter-Stachanows:

Ein wesentlicher Unterschied zwischen Mensch und Bot ist: Bots benötigen keinen Schlaf. Wobei jedoch der Bots-Schöpfer auf simple Weise den Tagesablauf eines Menschen simulieren kann. Es gibt Bots die durchaus auch als solche erkennbar sein können. Der bereits erwähnte „*de_philo_bot*“ sagt bereits im Namen deutlich welcher Natur er ist.
Man kann mit Vanellus analysieren, wie lange ein User aktiv ist. Als Beispiel wird ein Zeitintervall von 1 Stunde mit mindestens 2 Tweets pro Stunde genommen. Nachdem der Twitter-Server von 0:00 bis 2:00 CEST keine Tweets durchgibt, sind es maximal 22 Stunden/Tag oder 242 Stunden im Beobachtungszeitraum. Insgesamt 12 Bots arbeiten durch. Es sind dies in der Regel Bots von Radiostationen wie z.B. „*NowPlayingSRW3*“, die das gerade angespielte Lied tweeten. Auch die Bots der Zeitungen

wie z.B. „*handelsblatt*“ sind fast rund um die Uhr aktiv. Ein interessanter User ist der ebenfalls rund um die Uhr arbeitende „*FakeRedakteur*“. Dieser Bot baut aus den Textbausteinen der aktuellen Meldungen nach einem mathematischen Modell (Markov-Kette) neue zusammen. Z.B. „*Der frühere Fernsehstar Bill #Crosby ist in Washington mit US-Präsident Donald Trump vorbereitet ha...*“ (Der Tweet endet tatsächlich so). Der inzwischen zu einiger Berühmtheit gelangte Bot „*DeepDrumpf*“ verwendet ebenfalls diese Methode. Er baut aus den Tweets des Users „*RealDonaldTrump*“ neue zusammen.

Die Twitter-Sprinter:

Man kann sich auch die Peak-Tweetrate der User innerhalb einer Periode ansehen. Bei einer Periodenlänge von 15 Minuten hat dabei der eindeutig als Bot identifizierbare User *ffd365* mit 155 Tweets die Nase vorne. *ffd365* spuckt dabei – im Durchschnitt – alle 5,8 Sekunden einen Tweet aus. Interessant ist die Nr. 2 *AfDwirkt*. Das ist auch ein von der AfD gerne verwendeter Hashtag. Grafik 11 zeigt das zeitliche Verhalten von *AfDwirkt* mit einer Periode von 1 Minute. Die Spitzenleistung sind 13-Tweets/min. Es hat sich jemand am Fr. 4. Mai um 3:15 (CEST) hingesetzt und eine Viertelstunde lang was das Zeugs hält Retweets gesendet. Nach ca. 10 Minuten fällt seine Leistung bereits deutlich ab. Es gibt ansonsten keinerlei Tweets dieses Users. Es kann sich auch um den Testlauf eines Bots gehandelt haben. Vanellus erkennt jedoch einen reinen Namenswechsel eines Users. Der User wurde mit Sicherheit nicht unter einem anderen Namen wieder aktiv. Aber man kann sich bei Twitter als neuer User anmelden. Vermutlich hat jemand in einem überdrehten Zustand eine Twitter-Salve abgefeuert und hatte anschließend genug von der Sache.



Grafik 11: Histogramm von *AfDwirkt* mit 1 min. Zeitintervall.

Namensänderungen:

Ein User kann wie erwähnt seinen Namen jederzeit und auch mehrmals ändern. Es bleibt jedoch die von Twitter vergebene User-Id (eine 64-Bit Zahl) gleich. Man kann sich in Vanellus diese Namens-änderungen und auch die Anzahl der Tweets mit alten und neuen Namen anzeigen lassen. Es machen überwiegend User – vorwiegend Bots - mit eher sexuell anrüchigen Texten davon Gebrauch. Twitter kann natürlich genauso leicht wie Vanellus mitverfolgen wie der User nun heißt. Auf Grund der Analyse habe ich jedoch den Eindruck, dass Twitter gegen Bots im Grunde nichts hat. Mehr User, mehr Tweets sind in der Bilanz ein Zeichen von Erfolg. Twitter geht gegen Bots offensichtlich nur vor, wenn diese in das Kreuzfeuer der Kritik kommen und das Image der Firma beschädigen. Es umgeht auch

Vanellus die Daten-Beschränkungen für einen freien Zugang (siehe dazu den technischen Teil). Twitter könnte und kann dies mit elementaren Methoden erkennen. Bisher hatte ich keine Probleme. Es könnte jedoch irgendeinmal so weit sein, dass Vanellus nur mit einem Daten-Abo weiterhin massiv Daten saugen darf.

Zusammenfassung und weitere Schritte:

Ich bin mir relativ sicher, dass es keinen weiteren – nur mit einem freien Zugang ausgestatteten – Bot gibt, der fast vollständig den Bestand aller deutschsprachigen Tweets herunterladen und in einer eigens dafür konzipierten, sehr effektiven Datenbank verwalten kann. Es sind auch die Analysefähigkeiten bereits relativ ausgefeilt und auch sehr effektiv implementiert. Es hat jede Software immer um genau ein Feature zu wenig. Aber in diesem Bereich sehe ich – leider - relativ geringen Handlungsbedarf. Leider insofern, da mir die Implementierung derartiger Features Spass macht. Sie sind auch eine interessante algorithmische Herausforderung. So dauert die Erkennung von Zwillingen bei 10 Millionen Tweets und 2,5 Millionen Usern auf meinem PC nur ein paar Sekunden. Direkt jeden User mit jeden anderen vergleichen würde mehrere Tage dauern. Man muss sich Tricks einfallen lassen, wie das eleganter und schneller geht.
Für dieses Analyse habe ich einen fixen Datenbestand verwendet und die Arbeit auf einem eigenen PC durchgeführt, damit sich die Resultate nicht während des Schreibens laufend verändern. Man könnte sie jedoch genauso in Echtzeit parallel zum Download-Betrieb durchführen.

Ein bisher noch nicht behandeltes Thema ist die Sentiment Klassifizierung von Tweets bzw. von Usern. Wie bereits erwähnt bieten sich dazu die klar positionierten Freunde und Feinde der AfD an. Es sollte jedoch auch für subtilere Fragestellungen eine Sentiment Analyse möglich sein.

Vanellus ist bisher ein sehr ungewöhnlicher Kiebitz. Er hält sein Maul. Man könnte auch auf diesem Gebiet arbeiten. Allerdings müsste man zu diesen Zweck erst definieren, was der Kiebitz in welchen Sinn sagen soll. Dazu habe ich im Moment keinerlei Vorstellungen. Möglicher Weise fällt dem Leser dieser Zeilen dazu etwas ein.

Möglich ist auch eine Erweiterung der Aktivitäten auf andere soziale Medien – insbesondere auf Facebook. Ich habe mit Twitter begonnen, weil die Twitter-Kommunikation relativ einfach gestrickt ist und die Unterstützung durch Programmbibliotheken sehr gut ist. Letzteres sollte jedoch auch bei Facebook der Fall sein.

Teil II: Ein Blick unter die Vanellus Motorhaube:

Vanellus ist in JavaFX-8 implementiert. Als Twitter-Library wird twitter4J verwendet. Diese Library ist ausgereift und stabil. Die generelle Architektur – inklusive einer Reihe von Klassen wie z.B. für die Datumsbehandlung – wurde von meinem Trading-Bot „*Cashbot*“ übernommen. Teilweise steckt aber auch Know-How aus den Zeiten des Schachmonsters Hydra drinnen.
Die wesentlichen Bestandteile sind ein User-Interface inklusive eines Chart-Moduls, eine einfache Makro bzw. Kommandosprache, ein eigener periodisch aufgerufener Task zur Abfrage und Einlesen der Daten vom Twitter-Server, ein Task der zur festgelegten Zeiten die eingelesenen Daten auf die Festplatte schreibt, eine eigene Datenbank, eine Sammlung von sehr effektiven Collections und einem Analyse-Modul der die im 1. Teil präsentierten Resultate aus der Datenbank berechnet. Es werden die angezeigten Daten streng getrennt von den gespeicherten Daten gehalten (Model-View-Controller Konzept). Ein eigener Task transferiert die Model-Daten periodisch in den View-Bereich. Dieser Task

stellt sicher, dass sich die Twitter-Lese-Tasks nicht mit der internen GUI-Loop in die Haare geraten. Der Versuch Daten vom Twitter-Server direkt anzuzeigen erzeugt mit Sicherheit einen Crash.
Anmerkung: Bei CashBot werden die Daten vom Broker-Server geladen. Die generelle Problematik ist identisch.

Die Kommandosprache:

Das untenstehende Kommando zeigt im Display-1 alle Tweets von Alice_Weidel und Beatrix_vStorch an, die im Text die Substrings „AfD“, „Weidel“ oder „Storch“ nicht jedoch den Substring „Gauland“ enthalten.

Die meisten Parameter besitzen Defaultwerte. Wenn man z.B. aus den Tweets aller User selektieren will, dann lässt man den Parameter `users` weg. Genauso kann man die `query` weglassen, wenn alle Tweets der beiden User angezeigt werden sollen. Man kann auch mit `since:„2018-05-01 06:00“` und `until:„2018-05-01 07:00“` den Abfragezeitraum auf den 1. Mai von 6 bis 7Uhr in der Früh einschränken. Wenn man den gesamten Zeitraum lesen will, lässt man `since` und `until` weg.

Man kann wie in einer Shell in der Liste der bisher verwendeten Kommandos scrollen, diese editieren und erneut verwenden. Man kann regelmäßig vorkommende Kommandos und im speziellen Kommandoabfolgen in einem File mit der Endung „.vn“ speichern und dieses direkt abrufen. Z.B. wurden die Kommandos zum Laden der Datenbank im File „readDays.vn“ gespeichert. Mit `readDays;` wird dieser Befehl ausgeführt.
Mit dieser Kommandosprache können auch die Tasks zum Einlesen der Twitter-Daten und zum periodischen Abspeichern in der Datenbank gestartet und gestoppt werden. Die Kommandosprache ist ausbaufähig, weil sie bisher keine Platzhalter und Kontrollstrukturen unterstützt. Platzhalter wären (auch in Cashbot) sinnvoll, der Nutzen von Kontrollstrukturen ist fraglich. Eine eigene CashBot bzw. Vanellus Programmiersprache wäre jedoch ein nettes – und nicht allzu schwieriges – Projekt.

„Normale“ Benutzer können mit dieser Form der Kommandoeingabe meistens wenig anfangen, als alter Hacker der mit Unix groß geworden ist ziehe ich eine Kommandosprache/Shell eindeutig vor.

Die Datenabfrage:

Die Java twitter4J Library stellt zwei Schnittstellen zur Datenabfrage vom Twitter-Server zur Verfügung. Mit der Klasse `Twitter` schickt man an den Server eine Query und erhält bis zu 100 Tweets die diese Query erfüllen. Mit dem freien Zugang kann man in 15 Minuten bis zu 180 Queries schicken. Wenn man über dieser Quote ist erhält man eine `rate exceeded` Exception und muss einen 15-minütigen Boxenstopp einlegen. Nach meinen Erfahrungen ist das 180 Query Limit nicht exakt. Manchmal werden auch bis zu 250 Abfragen toleriert, manchmal muss man bereits bei 160 an die Box. Bei 150 Abfragen ist man auf der sicheren Seite. Man kann damit alle 6 Sekunden eine Abfrage schicken und erhält maximal 1.000 Tweets/Minute. Praktisch ist es weniger, weil die Tweets nicht gleichmäßig daher kommen. Wie man aus Grafik 1-3 sieht, kann man damit nicht alle deutsch-sprachigen Tweets erfassen.
Mit der Klasse `TwitterStream` braucht man keine explizite Abfrage stellen. Der Twitter-Server liefert sofort jeden Tweet der der Query entspricht. Allerdings ist dieses Interface noch wesentlich stärker gedrosselt und die Implementierung in twitter4J nicht besonders stabil. Es wird daher ausschließlich Polling mit Hilfe der `Twitter` Klasse verwendet. Auch CashBot verwendet diese Methode zur Abfrage der Kurse an der Börse in Chicago. Das Problem ist nicht gänzlich trivial, da man es Thread-Save implementieren muss. Man darf u.A. wie bereits erwähnt Daten nicht direkt an das GUI

schicken. Wenn man mehrere Reader-Tasks gleichzeitig laufen lässt, dürfen sich diese beim Abspeichern der Daten nicht gegenseitig überschreiben.

Das Problem des Datenlimits wurde durch die Anmeldung von in Summe 4 Bots gelöst. Wobei 2 Bots auf den User Vanellus und 2 auf die Userin Amalia_vEnz angemeldet sind. Für das Einlesen werden 3 Bots verwendet, die sich jeweils im Takt von 2,1 Sekunden abwechseln. Der 4 Bot dient zum Versenden von Test-Tweets sowie zu zusätzlichen Abfragen wie z.B. die Liste der Follower. Die Bots laufen unter einer gemeinsamen Oberfläche. Man kann sehr leicht zusätzliche Bots hinzufügen (würde aber wohl irgend einmal Schwierigkeiten mit Twitter bekommen). Die Bots haben keine hart-kodierte Aufgabe. Man kann z.B. beim Starten des Abfrage-Tasks festlegen, welche Bots für das Einlesen der Daten zuständig sind.
Es bleibt damit jeder der drei Bots unter 150 Abfragen in 15 Minuten. Die maximale Rate sind 2,850 Tweets/min. Das reicht für das normale Datenvolumen von gut 1,000 Tweets/min. Bei Real gegen Bayern wurde wahrscheinlich der eine oder andere „Tooor“ Tweet verschluckt. Vanellus hat jedoch das generelle Verhalten sehr gut erfasst.
Der Twitter-Server meldet manchmal „Server overloaded, try later“. Zwischen 0:00 und 2:00 CEST werden Requests überhaupt ignoriert. Das ist höhere Gewalt, ebenso wie die gelegentlichen Aussetzer meiner Hochland-Internetverbindung. Twitter garantiert generell nicht, dass man vom Server alle Daten bekommt (egal wie viel man für ein Abo zahlt). Eine derartige Garantie wäre auch nur mit extremen Aufwand zu erfüllen. Tweets sind keine Transaktionen in einer Bankdatenbank. Es ist geht nicht sehr viel verloren, wenn eine geringe Rate von Tweets unter den Tisch fällt.

Ursprünglich war ein Tweet auf 140 Zeichen beschränkt. Dies wurde inzwischen auf 280 Zeichen erweitert. Die offizielle stabile Version 4.04 von twitter4J unterstützt nur 140 Zeichen. Es gibt jedoch eine neuere Version 4.06 in der man einen extended Parameter setzen kann. Damit wird auch das neue Format unterstützt. Manche Retweets werden jedoch bei 140 Zeichen abgeschnitten. Das dürfte nicht an twitter4J, sondern an der obsoleten Twitter-App des Benutzers liegen. Ein Teil der Retweets kommt in voller Länge an.

Die eingelesen Daten werden in der In-Memory-Datenbank gespeichert. Zuvor werden beim Tweet-Text eine Folge von whitespace-Characters durch ein einziges ersetzt und linefeeds entfernt. Ein eigener Task schreibt die Datenbank kurz nach Mitternacht CEST (zu diesem Zeitpunkt ist der Twitter-Server außer Betrieb) auf die Festplatte. Die Datenabfrage läuft im Moment auf einem eigenen Rechner. Die In-Memory- Datenbank wird nach der Write-Operation gelöscht. Auf diese Weise erzeugt man jeweils einen Snapshot für einen gesamten Tag. Will man z.B. wie oben 11 Tage behandeln, dann liest man diese 11 Snapshots hintereinander ein. Durch eine neuerliche Write-Operation speichert man die gesammelten Daten in einer einzigen Datenbank. Man kann auch weiterhin die Datenbanken Tag für Tag einlesen. Die 11-Tage Datenbank ist jedoch etwas kompakter, da z.B. Tweet-Texte die über mehrere Tage wieder verwendet werden nur einmal abgespeichert bzw. gelesen werden müssen. Die In-Memory Datenbank ist jedoch in beiden Fällen identisch, da ein Text der mehrmals vorkommt nur 1x gespeichert wird.

Die Datenbank:

Vanellus hält alle Daten in einer selbst gebauten In-Memory-Datenbank. Die hereinkommenden Tweets werden in dieser Datenbank abgespeichert. Man kann die Datenbank auf der Platte sichern, muss sie aber zur Analyse zuvor wieder in das Memory einlesen. Eine Datenbank die zum Teil im Speicher, zum Teil auf der Platte verwaltet wird, wird ziemlich komplex. Außerdem hat man de facto „verloren“, wenn man bei der Analyse von der Platte lesen muss.

Es wird von einem Tweet die Twitter-UserId, der Zeitstempel, die Tweet-Id, der User-Name, der Tweet-Text und die Abfrage auf Grund dessen er selektiert wurde, gespeichert. Es wird ein Tweet in mehrere Tabelle aufgeteilt. Es wird für den User-Namen und den Tweet-Text eine Hashzahl erzeugt. Nachdem die Java-String Hashberechnung Schrott ist, wird stattdessen die wesentlich bessere CRC32 Methode bzw. Klasse verwendet. CRC32 erzeugt wie der Name schon sagt eine 32-Bit Zahl. Allerdings liefert die Methode long (64-Bit) zurück und es wird auch intern eine long abgespeichert. Ich wollte mir bei der Umwandlung in int keine Probleme einhandeln. Außerdem könnte es sich herausstellen, dass CRC32 zu viele Hashkollisionen erzeugt. In diesem Fall könnte man – ohne die Datenformate ändern zu müssen – nur an einer Stelle den Hashkode auf eine 64-Bit Methode umstellen (und alte Datenbestände entsprechend neu berechnen).

Sowohl der User-Name als auch der Tweet-Text werden in eine Hashtabelle mit dem CRC32-Key eingetragen. Damit werden User-Namen und auch Tweet-Texte die sich wiederholen nur 1x gespeichert. Unmittelbar wird ein Tweet in einer Tabelle mit dem Record
`<Message-Id>,<Time>,<Query-Hash>,<User-Hash>,<Text-Hash>`
gespeichert. Wobei jedes Element eine Long ist. Die Query, der User-Namen und der Tweet-Text wird bei Bedarf aus den jeweiligen Hashtabellen gelesen. Zusätzlich gibt es noch eine Hashtabelle die die Twitter-User-Id mit der User-Id (die sich aus dem Hashkode des Users-Namens ergibt) abspeichert. Diese zusätzliche Tabelle ist nicht besonders elegant. Sie wurde erst später hinzu geflickt, nachdem ich das Problem des „Name-Changes“ erkannt habe. Sinnvoller Weise sollte der Record die Twitter-User-Id als 6 Element enthalten.
Der <Query-Hash> ist im Moment nicht besonders sinnvoll, da immer nur mit „lang:de“ abgefragt wird. Ursprünglich habe ich jedoch mit verschiedenen Querys abgefragt. Es könnte auch sein, dass dieses Feature wieder nützlich wird. Es ist generell nicht uninteressant, woher man einen Tweet hat. Die Datenbank speichert einen Tweet nur ein einziges Mal ab. Man kann daher von der Platte auch mehrere Datenbank-Snapshots mit sich überlappenden Tweets laden. Redundante Tweets werden ignoriert. Dasselbe gilt, wenn man mehrere Abfrage-Tasks mit verschiedenen Querys gleichzeitig laufen lässt. Auch hier können - wenn z.B. Task-A nach „AfD“ und Task-B nach „Orban“ abfragt - die selben Tweets gelesen werden. Dieses Problem stellt sich allerdings im Moment nicht, da generell nur eine Query „`lang:de`“ verwendet wird.
Ein Tweet benötigt intern in Summe etwa 120 Bytes. Die oben präsentierte Analyse verwendet etwas mehr als 10 Millionen Tweets. Das entspricht ca. 1,2 GByte Speicher. Für die Analyse werden zusätzlich Datenstrukturen im Bereich von einigen 100 MByte verwendet. Die Java VM allokiert per Default 1 Giga. Man muss daher an den Parametern der JVM drehen und einen PC mit mindestens 8GB, vorzugsweise jedoch 16GB besitzen.

Generell bin ich mit der Datenbank bisher sehr zufrieden. Ein nettes Feature wäre das „Ausputzen“ von irrelevanten Tweets z.B. der „*Arikatytm31*“ Hashtag Gang. Es sollte eine Blacklist geben auf Grund derer die Tweets von diesen Usern ignoriert werden. Das würde in einem Aufwaschen auch das „Ausputzen“ Problem lösen. Man liest eine Datenbank ein, die Gang wird ignoriert und speichert sie wieder ab. Die Gang ist weg. Wahrscheinlich bringt das wesentlich mehr als an Feinheiten der Datenbank drehen zu wollen.

Die Collections:

An und für sich ist die Standard-Library von Java mit einer umfangreichen Kollektion von Collections ausgestattet. Mein Freund und damaliger Teamkollege Steffen hat daher den Kopf geschüttelt, als ich angefangen habe, in CashBot für die Speicherung von Börsen-Zeitreihen eine eigene Collection zu implementieren. Wir halten noch immer aus Gewohnheit und zum Erhalt der Freundschaft regelmäßige

Skype Sitzungen ab. Steffen schüttelt noch immer den Kopf, wenn ich ihm von meiner neuesten Collection Klasse erzähle, die die Analyse in Vanellus nun super-schnell und überhaupt erst möglich gemacht hat. Er meint, das wäre auch mit dieser und jener Standardlibrary gegangen. Obwohl wir uns schon wie ein altes Ehepaar immer mehr angleichen: In dieser Frage werden wir wohl nie eins werden. Genauso wie sich Anni und Chrilly nie einigen werden, ob es nun „*Viertel über Fünf*“ (Chrilly) oder igitt „*Viertel Sechs*“ (Anni) ist. Der kleinste gemeinsame Nenner ist 17:15.

Aus Sicht des alten Hackers haben die Java-Collections mehrere Nachteile. Erstens wurden sie nicht von ihm geschrieben. Darüber hinaus wird in einer Collection eine Integer bzw. Long-Zahl nicht als solche, sondern als Integer/Long-Objekt gespeichert. Statt 4 bzw. 8 Bytes benötigt dies ein vielfaches. Die CashBot bzw. Vanellus Collections speichern hingegen die Werte in einer int/long Array ab. Es fällt nur der minimale Array-Overhead an. Dafür muss jedoch ein Record in mehreren Arrays abgespeichert werden und wenn eine Collection nicht nur aus Long und Int, sondern aus 2 Long und 1 Int besteht, muss eine neue Klasse geschrieben werden.
Neben dynamisch Arrays sind Hashtabellen das Kernstück der Collections Klasse. Man kann das Problem der Hash-Kollision durch eine Liste von Records lösen. Vanellus verwendet jedoch das Prinzip von Schachdatenbanken. Es wird in der Tabelle der nächste freie Slot verwendet. In Schach löst man das Problem des Überlaufes indem man ältere und nicht so wichtige Einträge überschreibt. Das ist bei der Twitter-Analyse nicht möglich. Falls die Anzahl der Einträge mehr als 50% der Tabellen-Größe übersteigt, wird die Tabelle verdoppelt. Bei einer Füllung bis 50% ist die Anzahl der Runs bis zum nächsten freien Slot noch relativ gering. Die Run-Länge wächst jedoch mit steigenden Füllgrad steil an. Diese Methode ist sehr effektiv, man muss jedoch die Größe der Hashtabelle vorab relativ gut abschätzen können. Im Schach gilt hingegen die einfache Regel. Die Tabelle ist so groß wie möglich. Ist die ursprüngliche Tabelle zu klein, dann wird mehrmals beim Verdoppeln der Tabellengröße komplett rehashed. Ist sie zu groß kann man u.A. an die Grenzen des JVM-Heaps stoßen. Außerdem will man am Ende der Operation in der Regel alle gültigen Elemente abfragen. Dazu muss man die gesamte Hashtabelle abklappern und erhält – bei zu großer Tabelle – meistens die Information „kein Eintrag an dieser Stelle“.

Die handgeschnitzten Hashtabellen enthalten auch spezielle, nicht-Standard Operation wie z.B. setze das Minimum/Maximum von aktuellen und bisherigen Wert. Oder erhöhe einen Counter wenn der Key bereits vorhanden ist. Das lässt sich auch mit der Standard-Library implementieren. Allerdings nicht als einziger simpler und effektiver Methodenaufruf. Neben Dynamischen Arrays und Hashtabellen enthalten die Collections auch einen Heap. Dieser wird u.A. dazu verwendet um etwa die 100 häufigsten Hashttags auszugeben. Der Heap ist natürlich ebenfalls als dynamische Array implementiert. Im Moment sind keine Erweiterungen der Collections geplant.

Ein Stachel im Fleisch des alten Hackers ist die Verwendung der Java-String Klasse. Ein Java-String schleppt ziemlich viel overhead mit sich herum. Das ergibt sich schon aus der Tatsache, dass er – wie alles in Java – ein Objekt ist. Es juckt mich zwar schon länger dies durch eine wesentlich effektivere Klasse zu ersetzen. Letztendlich würde es darauf hinauslaufen eine Reihe von anderen Klassen neu zu schreiben, da diese ebenfalls Strings als Parameter bzw. Return-Werte verwenden. Möglicher Weise ist eine derartige Klasse auch bereits erfunden worden. Es scheint sinnvoller den Aufwand für die Sentiment-Analyse von Vanellus zu verwenden.

Danksagung:
An Amalia_vEnz, dass sie sich extra für Vanellus einen Twitter-Account zugelegt hat. Mein Dank gilt auch Steffen. Auch wenn er bei den Collections den Kopf schüttelt, habe ich von ihm eine Reihe von wertvollen Hinweisen erhalten.