Generierung von fiktiven Donald Trump Tweets

Michael Bierschneider
Universität Regensburg
Michael.Bierschneider@stud.
uni-regensburg.de
Matr. Nummer: 1652774

Christoph Whürl
Universität Regensburg
Christoph.Wuehrl@stud.
uni-regensburg.de
Matr. Nummer: 2127095

Maximilian Weissenbacher
Universität Regensburg
Maximilian.Weissenbacher@stud.
uni-regensburg.de
Matr. Nummer: 2052821

Spätestens seit Bekanntgabe seiner Kandidatur für das Amt des Präsidenten der Vereinigten Staaten von Amerika polarisiert Donald J. Trump mit seinen Tweets auf dem Nachrichtendienst Twitter – weltweit! Auf seinem Kanal folgen ihm inzwischen 85,3 Millionen Menschen, um seine durchschnittlich 33,2 Tweets pro Tag zum täglichen politischen Geschehen zu verfolgen.[1]*Aus den mittlerweile mehr als 50.000 Tweets ergibt sich ein äußerst interessanter Datensatz für das Natural Language Engineering.*

Dieses Projekt befasst sich mit der Generierung von neuen, zufälligen Tweets auf Basis der Tweets von Donald Trump. Diese generierten Tweets sollen denen von Donald Trump in Form und Sprachstil ähneln. Hierfür wurde ein Bot programmiert (fortan Trumpbot), welcher ein oder mehrere Topics aus einer zuvor definierten Liste akzeptiert. Aus den erkannten Topics kann der Trumpbot, mittels der Verwendung von Markov-Ketten, Tweets generieren. Um die Glaubwürdigkeit von generierten Tweets zu überprüfen, werden zufällig echte oder generierte Tweets zurückgegeben. Daraufhin soll der Nutzer abschätzen, ob es sich bei dem ausgegebenen Tweet um einen echten oder einen Fake Tweet handelt. Anschließend erhält der Nutzer eine Bestätigung, ob seine Vermutung richtig war oder nicht.

Index Terms*- NLP, Tweet Generation, Donald Trump, Clustering, RegEx*

1 Lösungsidee

Als Lösungsansatz, um neue Tweets zu erzeugen, soll hier die Markow-Kette verwendet werden, um anhand von vorher erzeugten N-Grammen neue Tweets generieren zu können. Damit diese Markov-Ketten besser an das entsprechende Topic angepasst sind, erscheint eine vorherige Gruppierung der Trump-Tweets sinnvoll. Für diese Gruppierung sollen zwei unterschiedliche Verfahren betrachtet werden, einerseits das Finden von *Clustern* im vorhandenen Datensatz, welchen dann passende Themen zugeordnet werden und andererseits das Herausfiltern von Tweets, die ein gleiches Thema behandeln, mittels der Suche mit *Regular Expressions*. Um dann das richtige Markov-Modell je nach Thema auszuwählen, soll nach diesen unter der Zuhilfenahme von *Regular Expressions* in der Eingabe des Benutzers gesucht werden.

2 Tweetgenerierung

Um die Tweet-Generierung möglich zu machen, wurden zwei verschiedene Ansätze verwendet, die unter diesem Punkt beschrieben werden. Um beide Ansätze realisieren zu können, waren zunächst Vorverarbeitungsschritte nötig. Zuerst wurde ein Datensatz aus dem *Trump Twitter Archive* heruntergeladen.[2] Im *Trump Twitter Archive* finden sich nahezu alle Tweets, die Donald Trump in seiner Amtszeit als Präsident der Vereinigten Staaten verfasst hat, weshalb sich dieser Datensatz gut als Basis unserer Tweet Generierung eignet. Der Datensatz für dieses Programm wurde im April 2020 heruntergeladen, weshalb beispielsweise Tweets zum Thema *Covid-19* bereits enthalten sind. Tweets über die *Black Lives Matter-Bewegung* sind aber aufgrund des späteren Ereignisdatums noch nicht im Datensatz.

2.1 Vorverarbeitungsschritte

Um den Datensatz sinnvoll nutzen zu können, musste dieser erst präpariert werden. Unter anderem werden dabei Links und Bilder entfernt, da diese keine relevanten Sprachdaten von Donald Trump enthalten. Im Datensatz ist für dieses Programm nur der Text der Tweets relevant, weshalb andere Spalten, wie die Tweet-ID oder die Quelle, entfernt wurden. Um den Sprachstil von Donald Trump nicht zu verfälschen wurden Retweets nicht berücksichtigt, sondern nur Tweets, die Donald Trump selbst

verfasst hat. Außerdem wurden mit Hilfe der erstellten Funktion „clean_tweets“ die einzelnen Tweets gesäubert. Beispielsweise wurden Sonderzeichen durch Leerzeichen ersetzt, aus „#“ wurde „ # “, damit bestimmte Wörter nicht zwingend an dem Hashtag gebunden sind. Nach dem Säubern der Tweets und dem Entfernen von leeren Tweets waren 38098 verschiedene Tweets von Donald Trump übrig. Diese Tweets wurden für die beiden Ansätze, dem *Topic Clustering mit GloVe-Embeddings/TF-IDF* und der *Topic Generation* mit regulären Ausdrücken verwendet, welche im Folgenden genau erläutert werden.

2.2 Topic Clustering

Mit einer Clustermethode sollen ähnliche Tweets zu Gruppen zugeordnet werden. Um dies zu ermöglichen müssen die Tweets zunächst in einem Vektorraum dargestellt werden. Hierfür wurden sowohl *GloVe-Embeddings* als auch *TF-IDF-Vektoren* verwendet, um zu überprüfen, mit welchem Ansatz bessere Cluster zu erreichen sind.

2.2.1 GloVe-Embedings

Um die Tweets als Vektoren darzustellen, wurden die Glove-Twitter-Embeddings geladen.[3] Die Liste an Tweets in “clean_tweets” wird dann in *Glove-Embeddings* umgewandelt. Falls ein Wort von einem Tweet im Embedding-Index enthalten ist, wird sein Embedding zu einer Liste hinzugefügt. Falls ein Wort nicht im Embedding-Index enthalten ist, wird es nicht zu der Liste hinzugefügt. Als Ergebnis ist die Anzahl an Embeddings: 777853. Dazu kommen 67873 Null-Embeddings, was 8,73% entspricht. Anschließend wurde der durchschnittliche Embedding-Wert für jeden Tweet ermittelt und in der Liste „embed_Tweets“ gespeichert. Die Liste „embed_Tweets“ enthält also 38098 verschiedene Tweets, wobei jeder Tweet aus einem Array aus 200 Dimensionen besteht, da die GloVe-Embeddings diese Zahl vorgeben. Mittels der Hauptkomponentenanalyse (PCA), einer Methode der multivariaten Statistik, ist es möglich, die 200 Dimensionen eines jeden Tweets auf eine Dimension zu reduzieren, was für das spätere Gruppieren der Tweets von Vorteil ist. Als Ergebnis erhält man die Liste „pca_tweets“, in der nun jeder Tweet nur noch von einer Zahl repräsentiert wird. Dies ermöglicht es, die Tweets zu gruppieren, basierend auf der Zahl, die sie vertritt.

2.2.2 Tf-IDF

In einem anderen Ansatz sollen die Tweets als Vektoren mit berechneten TF-IDF-Werten dargestellt und geclustert werden. Jeder Tweet wird dabei durch einen n-dimensionalen Vektor repräsentiert, wobei n die Gesamtzahl aller im Korpus vorkommenden Wörter ist. Diese Dimensionen werden mit den TF-IDF-Werten der im Korpus vorkommenden Wörter in Bezug auf den jeweiligen Tweet gefüllt. Bei TF-IDF handelt es sich um ein Produkt, welche sich aus der *term frequency*, wie häufig ein Wort in einem Dokument auftritt, und der *inverse document frequency*, in wie vielen Dokumenten des Korpus ein Wort überhaupt auftritt, zusammensetzt. Die *term frequency* steigt, wenn ein Wort besonders häufig in einem Dokument auftritt und die *inverse document frequency* steigt, wenn dieses Wort besonders selten in Dokumenten vorkommt. Ein Dokument wird hier als einzelner Tweet definiert. Zur konkreten technischen Umsetzung wurde das *sklearn-Package* verwendet, im speziellen wurde auf das Modul *TfidfVectorizer* zurückgegriffen.[4] Diesem kann mit der Funktion „.fit_transform()“ eine Liste der gesäuberten Tweets übergeben werden, welche als Liste von Vektoren mit TF-IDF-Werten zurückgegeben wird. Anschließend wurde diese Liste durch die „.todense()“ Methode in eine Matrix umgewandelt, um die weitere Bearbeitung zu ermöglichen. Auf diese Matrix konnte dann die Hauptkomponentenanalyse (PCA) angewendet werden, um das Clustern zu vereinfachen.

2.2.3 K-Means Clustering

Mittels des K-Means-Algorithmus ist es möglich aus einer Menge von ähnlichen Objekten eine bestimmte Anzahl an Gruppen zu bilden. Die jeweiligen Gruppen oder auch *Cluster* sollen beim Trumpbot ein „Über-Thema“ definieren, indem sich Begriffe beziehungsweise Tweets mit ähnlicher Bedeutung befinden. Da der K-Means-Algorithmus besser mit Gruppen funktioniert, deren Größe ähnlich ist, wurden vier verschiedene Gruppen gebildet, welche in „labels“ gespeichert wurden. Mittels der erstellten Funktion „most_common_words“ wurden die am häufigst vorkommenden Wörter jeder Gruppe herausgegeben. Dabei wurden Stopwords und Wörter ohne Bedeutung entfernt, damit klarere Begriffe für jede Gruppe zurückgegeben werden. Im Optimalfall können durch das reine Betrachten dieser Wörter Über-Themen für die jeweiligen Gruppen definiert werden. Abbildung 1 und 2 zeigen die Top 10 Wörter der GloVe- und der TF-IDF-Methode.

```
Gruppe  1 :
['obama', 'democrats', 'fake', 'china', 'deal', 'nothing', 'states', 'media', 'united', 'border']
Gruppe  2 :
['interview', 'enjoy', 'course', 'golf', 'join', 'interviewed', 'poll', 'barackobama', 'obama', 'gop']
Gruppe  3 :
['birthday', 'ivankatrump', '“trump', 'poll', 'gop', 'cnn', 'interview', 'nbc', 'golf', 'billmaher']
Gruppe  4 :
['obama', 'barackobama', 'hillary', 'american', 'jobs', 'night', 'world', 'apprentice', 'house', 'china']
```

Fig. 1. Top 10 Wörter der GloVe-Methode

```
Gruppe  1 :
['republican', 'night', 'poll', 'business', 'god', 'look', 'honor', 'sir', 'proud', 'down']
Gruppe  2 :
['obama', 'democrats', 'fake', 'china', 'border', 'media', 'deal', 'world', 'states', 'united']
Gruppe  3 :
['maga', 'kag2020', 'poll', 'americafirst', 'carolina', 'florida', 'honor', 'iowa', 'hampshire', 'pennsylvania']
Gruppe  4 :
['obama', 'interview', 'apprentice', 'night', 'golf', 'jobs', 'course', 'business', 'looking', 'congratulations']
```

Fig. 2. Top 10 Wörter der Tfldf-Methode

Da diese 10 Begriffe einer jeden Gruppe noch keine klaren Topics herausfiltern, wurden zudem noch die kompletten Tweets von jeder Gruppe betrachtet. Dabei fiel auf, dass das Clustering mittels der GloVe-Methode besser funktionierte als mit der TF-IDF-Methode. Während die gruppierten Tweets der TF-IDF-Methode keinen wirklichen

thematischen Zusammenhang herauskristallisierten, konnte bei den Tweets der GloVe-Methode Überthemen für die jeweiligen Gruppen gefunden werden. „Politics“ zum Beispiel gilt als Über-Thema für Gruppe 4 der GloVe-Methode. In nächsten Schritt wurden die jeweiligen Gruppen mit ihren Tweets als txt-Datei gespeichert, damit sie für die Tweet-Generation mittels des *Markovify-Packages*, welches in Punkt 3 erläutert wird, verwendet werden können.

2.3 Topic Generation mit regulären Ausdrücken

Ein weiterer Ansatz, um Topics aus dem Datensatz herauszufiltern ist, dies mit regulären Ausdrücken zu erledigen. Mit der erstellten Funktion „find_topics(neue_liste, gesuchtes Wort‘)“ wird die Liste „clean_tweets“ mit den gesäuberten Tweets auf Wörter untersucht, die ein Topic darstellen sollen. Wird das gesuchte Wort in einem Tweet gefunden, wird der jeweilige Tweet an „neue_liste“ hinzugefügt. Beim Beispiel in Abbildung 3 wird das Topic „Corona“ deklariert. Dabei wird „clean_tweets“ auf folgende Wörter untersucht: ”flu, corona, covid, virus, pandemic, chinese virus”. Falls in einem Tweet aus „clean_tweets“ eines der Wörter vorkommt, wird der komplette Tweet an die anfangs noch leere Liste „covid“ gehängt. Die Ergebnisse werden zur besseren Übersicht in „topic_corona“ gespeichert. Zu dem Topic „Corona“ wurden 195 verschiedene Tweets gefunden.

```
covids = find_topic(covid,"flu|corona|covid|virus|pandemic|chinese virus")
```

Fig. 3. Topic Corona

Insgesamt wurden 29 verschiedene Topics deklariert. Im IPython-Notebook „RegEx_Ansatz“ wird anschließend das Markov-Modell für die Tweet-Generation mittels des *Markovify-Packages* erzeugt. Im nächsten Punkt wird erklärt, wie dieses funktioniert. Um die verschiedenen Markov-Modelle zu jedem Topic exportieren zu können, werden diese in ein JSON-Format umgewandelt und mit der Methode „json.dump()“ als Textdatei gespeichert. Somit kann jedes Modell bei der Webanwendung einfach importiert werden.

2.4 Markov Ketten

Wie bereits beschrieben, soll zur Erzeugung der Tweets der Markov-Prozess auf vorher erstellte N-Grammlisten angewendet werden. Genauer ist damit gemeint, dass das Auftreten eines Worts nur von einer bestimmten Anzahl (N) vorheriger Wörter abhängig ist. Diese Häufigkeiten können aus den jeweiligen Tweet Gruppen ausgezählt werden, um dann aus einem gegebenen Status (einer N langen Wortgruppe) den nächsten Status (das darauf folgende Wort) ableiten zu können. Hier kann auf das Markovify-Package zurückgegriffen werden.[5] In diesem Package können aus Text Markov-Modelle für beliebige N-Grammlängen erstellt werden und anschließend neue Texte auf Grundlage dieser N-Gramme generiert werden. Konkret wird zunächst Text eingelesen, um daraus *Dictionaries* aus Wortgruppen und Folgewörtern mit ihrer Häufigkeit zu erstellen. Anschließend wird bei der Erstellung von Text für jedes neue Wort, unter Betrachtung der vorherigen Wortgruppe der Länge N, aus allen möglichen Folgestatus zufällig das nächste Wort ausgewählt, wobei die Häufigkeiten hier mit in Betracht gezogen werden. Durch den Zufallsaspekt wird verhindert, dass aus demselben Startwort immer der gleiche Tweet generiert wird, allerdings werden häufiger vorkommende Folgewörter wahrscheinlicher ausgewählt. Des Weiteren erfolgt durch das Package auch eine Überprüfung, sodass sichergestellt wird, dass nicht ein bereits existierender Tweet generiert wird. Hier wurden die Defaultwerte des Package verwendet, womit ein generierter Tweet abgelehnt wird, wenn er einen Orginaltweet mit exakt 15 Wörter oder 70% der Wortanzahl des Satzes überlappt. In diesem Projekt wurden die in den txt-Dateien vorhandenen Tweets zunächst durch Regex an den vorhandenen *newline character* getrennt und in Listen gespeichert, um die einzelnen Tweets zu erhalten. Anschließend wurde für jede einzelne Themengruppe ein Markov-Modell erstellt. Durch mehrfache Versuche der Tweeterzeugung war abzusehen, dass die sinnvollsten Ergebnisse hier durch Trigramme erreicht werden können. Mit diesen Modellen konnten dann je nach Thema durch den Befehl „.make_short_sentence()“ entsprechende Tweets erzeugt werden, welche nicht länger als die bei Twitter üblichen 140 Zeichen waren. Während diese Methode für die mit Regex gefilterten Tweetgruppen zufriedenstellend funktionierte, ergab sich für die geclusterten Tweetgruppen das Problem, dass diese Cluster zu viele verschiedene Themen enthielten und die erstellten Tweets so selten dem geforderten Thema entsprachen. Um dieses Problem zu lösen, wurde auf die „.make_sentence_with_start()“ Methode des Markovify-Package zurückgegriffen. Dabei wird ein Startwort übergeben, mit welchem der Markov-Prozess durchlaufen wird. Hier wird bei den Modellen der geclusterten Tweetgruppen das gefundene Thema als Startwort übergeben, um zu garantieren, dass ein thematisch korrekter Tweet erstellt wird. Zwar ergaben sich so Limitationen, da der erzeugte Tweet immer mit einem festgelegten Wort begann und durch das Markov-Modell nur entsprechende Tweets mit dem Startwort verwendet wurden, allerdings überwog hier Überlegung der Sicherstellung der thematischen Korrektheit, sodass dieses Vorgehen verwendet wurde.

3 Webanwendung

Um Daten für die Evaluierung zu erhalten, wurde eine Webanwendung realisiert. Hierzu wurde zunächst eine statische Website auf einer EC2 Instanz von AWS gehostet. Die Website ist unter folgender IP erreichbar: `http://firecat.red:8001/`. Der Code zur Implementierung ist über Github einsehbar.[6] Die Anwendung weißt den Nutzer über ein Popup erstmals darauf hin, wie die

Funktionsweise des Trumpbots ist. Über einen *Switch* links oberhalb des Chatfensters kann der User zwischen zwei Versionen der Backendimplementierung wechseln: – Topicclustering mit K- Means (lambda_function_alt.py) und Topic Generation mit Regex (lambda_function.py).[7] Der Nutzer kann in der Eingabezeile mit dem Befehl „!help“ abfragen, welche Topics zur Verfügung stehen – diese sind je nach Version des Trumpbots unterschiedlich! Die Anforderungen für die anschließende Nutzung und Generierung von Tweets ist folgende: Die Nutzereingabe muss in Englisch erfolgen – da der Datensatz nur in englischer Sprache verfügbar ist. Es muss mindestens eines der Topics, welche über den Befehl „!help“ einsehbar sind, enthalten sein. Der Input kann ansonsten in freier und natürlicher Sprache, oder über *Buzzwords* übermittelt werden. Nach dem Klicken auf den Senden-Button gibt der Trumpbot einen Tweet zurück. Der Nutzer kann nun entscheiden, ob es sich bei dieser Nachricht um einen echten, von @realDonaldTrump erstellten Tweet handelt oder ob es ein vom Trumpbot generierter Tweet ist. Denkt der Nutzer, dass es sich um einen echten Tweet handelt, so kann er dies mit der Eingabe „!real“ übermitteln. Andernfalls kann der User die Echtheit mit „!fake“ anzweifeln. Durch erneutes Senden des Inputs bekommt man eine entsprechende Reaktion. Die Website ist über ein AWS API Gateway mit dem Backend verbunden. Hierbei handelt es sich um eine AWS Lambda Funktion, womit der Code in der Cloud ausgeführt werden kann, ohne dass sich Gedanken bezüglich eines Servers gemacht werden müssen.[8] Die AWS Lambda Funktion erhält dabei folgende Argumente: Eingabe des Nutzers (user_input), Angabe über Echtheit des Tweets (fake_tweet) (vorherige Markierung des Trumpbots), sowie den letzten übermittelten Tweet (last_tweet). Wenn die Eingabe des Nutzers „!help“ ist, so wird die Liste an Topics zurückgegeben. Ist die Eingabe des Nutzers „!real“ oder „!fake“, so wird fake_tweet mit user_input verglichen und ausgegeben. Werden keine Befehle gefunden, wird die Eingabe des Users nach Topics mit RegEx durchsucht. Nach Ermittlung des Topics wird zu 50% entweder ein echter Tweet ausgesucht, oder ein neuer, fiktiver Tweet mit dem Markovify-Package erstellt. Die nötigen Markovify-Models und echten Tweets werden aus AWS S3 bei Bedarf geladen. Alle Interaktionen mit AWS Lambda werden in der Datenbank AWS DynamoDB für die spätere Evaluation abgespeichert. Dabei wird die Nutzereingabe, der zurückgegebene Tweet und die Echtheit gespeichert und zudem das Reaktionsergebnis des Users, ob der zurückgegebene Tweet echt ist oder nicht.

3.1 Topic Detection

Dem Nutzer ist es möglich den Input in freier und natürlicher Sprache dem Trumpbot zu übergeben. Allerdings ist die Funktionalität auf die englische Sprache limitiert, da es nur einen englischen Datensatz gibt. Die verfügbaren Topics werden dem Nutzer über den Befehl „!help“ zur Verfügung gestellt. In AWS Lambda wird der Input des Nutzers mit regulären Ausdrücken durchsucht. Hierbei wird nicht nur schlicht gegen das Topic *gematcht*, sondern auch Synonyme berücksichtigt, beziehungsweise das Wort auf den Wortstamm reduziert. Werden mehrere Themen im User-Input gefunden werden diese ausgezählt. Der Trumpbot entscheidet sich anschließend für das Topic mit den meisten Treffern. Bei Gleichstand wählt das Programm zufällig.

```
if(re.search(r'(usa|america.*|land|states)', word)):
    array_of_topics.append("america")
```

Fig. 4. Topicdetection

Die Topicdetection funktioniert für beide Versionen des Trumpbot gleich.

4 Evaluation

4.1 Evaluationsidee

Es stellen sich hier besonders zwei Fragen, nach denen die beiden Systeme evaluiert werden sollen. Erstens wie überzeugend beziehungsweise ununterscheidbar die von den Systemen erzeugten Tweets von echten Trump-Tweets sind. Zweitens welches der beiden Systeme häufiger solche „überzeugenden Tweets“ erstellt. Um zu evaluieren, ob Personen zwischen einen generierten Tweet und einen von Donald Trump geschriebenen Tweet unterscheiden können, soll das getestete System zufällig einen zum Thema generierten Tweet oder einen tatsächlich existierenden Tweet aus dem entsprechenden Themencluster ausgeben. Der Benutzer soll dann entscheiden ob es sich um einen echten oder generierten Tweet handelt. Anhand dieser Daten kann dann abgelesen werden, wie häufig erzeugte Tweets für echt gehalten, beziehungsweise als Fake erkannt werden und wie sich dies im Vergleich zu echten Tweets verhält. Da diese Evaluationsstrategie mit beiden Systemen durchgeführt wird, können so auch die Daten zwischen den beiden Systemen verglichen werden, um eine Aussage darüber treffen zu können, welcher Ansatz ein besseres Ergebnis liefert. Des Weiteren ist es hier auch sinnvoll, die entsprechenden Tweets zu den Bewertungen der Nutzer mit abzuspeichern, um hier eventuelle Muster erkennen zu können, an welchen generierte Tweets erkannt werden, damit die Limitationen dieser Ansätze besser verstanden werden können und mögliche Verbesserungen gefunden werden können. Es muss bei der Evaluation des ersten Ansatzes (mit Glove Embeddings und K-means clustering) beachtet werden, dass nur echte Tweets ausgegeben werden, welche auch mit dem jeweiligen Startwort beginnen mit welchem auch erzeugte Tweets beginnen. Ansonsten könnten Fake-Tweets automatisch an diesem Startwort erkannt werden.

4.2 Erhebung der Daten

Um Nutzerinteraktionen mit dem Trumpbot zu erhalten, wurde der Link zur Anwendung in Foren der Universität Regensburg gepostet. Es wurden keine demographischen Daten der Nutzer abgefragt, aber es ist anzunehmen, dass die

meisten Teilnehmer Studenten der Informationswissenschaft beziehungsweise der Medieninformatik sind, oder Mitarbeiter der Universität, wie etwa Dozenten.

4.3 Auswertung der beiden Ansätze

4.3.1 Auswertung K-Means-Clustering

Beim ersten Ansatz mit der Markov-Tweet-Generierung auf Basis von vorher geclusterten Topics wurden insgesamt 581 verschiedene Daten erhoben. Davon waren allerdings nur 178 Interaktionen der Nutzer mit dem Trumpbot verwertbar, da falsche Befehle in die Inputleiste eingegeben wurden. Beispielsweise wurden Topics eingegeben, die nicht zur Auswahl standen oder nach Erhalten des Tweets wurde keine "!fake"/"!real"-Eingabe gemacht, weshalb die Tweetüberprüfung nicht stattfinden konnte. Mögliche Gründe hierfür sind das Nicht-Beachten der Anleitung, oder das Testen, ob andere Themen doch funktionieren.

Version 1	kmeans		
	tweet generated	tweet real	summe
user correct	56	49	105
user incorrect	31	42	73

Fig. 5. K-Means Auswertung

Abbildung 5 zeigt einen Überblick, wie viele Tweets richtig oder falsch zugeordnet wurden und ob es sich dabei um einen generierten Tweet oder einen echten Tweet handelt. Das Ergebnis ist, dass 105 Tweets von den Nutzern richtig und 73 Tweets falsch eingeordnet wurden, was einem Verhältnis von 69,5% entspricht. Bei den 178 Interaktionen wurden 87 Fake-Tweets herausgegeben und 91 echte Tweets von Donald Trump. Von den generierten Tweets wurden 56 als Fake-Tweet erkannt und 31 für echt gehalten. Das heißt, dass die Nutzer bei 35,6% der generierten Tweets dachten, dass es sich um einen echten Tweet von Donald Trump handelt. Von den echten Tweets wurden 53,8% für echte Donald Trump Tweets gehalten. Ein Grund, warum diese Zahl nicht höher ist, ist die teilweise wirre Grammatik von Donald Trump in seinen Tweets, durch die der Nutzer womöglich dachte, dass es sich dabei um einen Fake-Tweet handeln muss. Ein weiterer Grund hierfür kann sein, dass die Struktur der Tweets nicht immer zu 100% der Struktur der echten Donald Trump Tweets entspricht. Durch das Säubern der Tweets wurden zum Beispiel, wie oben angesprochen, Hashtags verändert, oder Satzzeichen anders dargestellt. In Summe lagen die Nutzer bei diesem Ansatz zu 41% falsch und zu 59% richtig.

4.3.2 Auswertung RegEx-Ansatz

Beim zweiten Ansatz mit dem deklarieren von Topics mittels Regulären Ausdrücken wurden insgesamt 193 verschiedene Daten erhoben. Ein Grund, warum die Zahl hier deutlich kleiner ist als die Zahl beim ersten Ansatz ist, dass der Link dazu erst in den Semesterferien in die Foren gepostet wurde, während der Link des ersten Ansatzes in der Vorlesungszeit gepostet wurde, als die Arbeitsbereitschaft höher war. Von diesen 193 Daten waren 101 Daten verwertbar, aus denselben Gründen, die zuvor genannt wurden. Bei diesen 101 Interaktionen wurden 53 Fake Tweets zurückgegeben. Von diesen 53 Fake Tweets wurden 33 Tweets korrekt von den Nutzern als Fake Tweets entlarvt, was 62,2% entspricht. 20 der ausgegebenen Fake Tweets wurden für einen echten Tweet von Donald Trump gehalten. Also konnte der RegEx-Ansatz bei 37,7% der ausgegebenen Tweets den Nutzer täuschen. Es wurden außerdem 48 echte Tweets von Donald Trump zurückgegeben, wobei 29 Tweets korrekt als echter Tweet erkannt wurden. 19 echte Tweets von Donald Trump, also 39,5%, wurden für Fake Tweets gehalten. Dies kann wieder wie zuvor beschrieben mit der teilweise wirren Sprache von Donald Trump zusammenhängen. In Summe lagen die Nutzer beim RegEx-Ansatz zu 38% falsch und zu 62% richtig.

Version 2	regex		
	tweet generated	tweet real	summe
user correct	33	29	6
user incorrect	20	19	3

Fig. 6. RegEx Auswertung

4.4 Interpretation

Die beiden Ansätze weisen eine ähnliche Erfolgsquote auf. Während der K-Means-Ansatz 35,6% der Nutzer täuschen konnte, konnte der RegEx-Ansatz bei 37,7% der generierten Tweets den Nutzer glauben lassen, dass der ausgegebene Tweet ein echter Donald Trump Tweet war. Im Gegensatz zu echten Donald Trump Tweets, welche Erfolgsquoten von 53,8% bzw. 60,4%, erzielten, konnte der Trumpbot nur 35-38% erreichen, sodass hier noch ein Unterschied erkennbar ist. Zusammfassend kann behauptet werden, dass es für die Erfolgsquote des Projekts keinen großen Unterschied macht, welcher Ansatz verwendet wird. Beim RegEx-Ansatz ist allerdings ein großer Vorteil, dass die Topics zuvor frei wählbar sind. Beim K-Means-Clustering ist die Themenauswahl stark eingeschränkt, weshalb viele Themen von den Nutzern abgefragt wurden, die im Programm nicht exisierten und der Bot somit keinen Tweet erzeugen konnte. Beim RegEx-Ansatz hingegen konnten die meisten Topics der Nutzer verarbeitet werden.

5 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde untersucht, wie Tweets, die denen von Donald Trump ähneln sollen, mittels verschiedener Methoden generiert werden können. Als Korpus wurden dafür echte Donald Trump Tweets verwendet, die anschließend in Topics geclustert wurden. Zum einen fand das Clustern über Reguläre Ausdrücke statt und zum anderen über GloVe-Embeddings. Für die GloVe-Methode wurden mithilfe des K-Means-Algorithmus

verschiedene Gruppen erstellt, die jeweils einem Thema entsprechen. Bei beiden Ansätzen wurden anschließend Markov-Ketten verwendet, um Tweets für das jeweilige Thema zu generieren. Für die Veröffentlichung des Projekts wurde eine Webanwendung erstellt, bei der die Nutzer prüfen konnten, ob die ausgegebenen Tweets generiert sind oder nicht. Bei der Evaluation kam heraus, dass bei beiden Ansätzen ähnliche Tweets, mit einer ähnlichen Erfolgsquote erzeugt wurden.

Bei der Analyse von generierten Tweets ist aufgefallen, dass einige Tweets ab einer gewissen Länge einen Themenwechsel enthalten oder dass in einem Nebensatz falsche Pronomen verwendet werden. Diese Fehler sind mit diesem Ansatz jedoch sehr schwer bis gar nicht zu vermeiden. Ein womöglich besserer Ansatz wäre die Nutzung von neuronalen Netzwerken zur Tweet-Generierung. Als ein weiteres Problem stellte sich die Satzzeichensetzung der generierten Tweets heraus. Mittels des Markovify-Packages ist es schwierig, Satzzeichen wie Kommas, Anführungszeichen oder Fragezeichen in einen sinnvollen Kontext zu setzen.

Für das Projekt wäre es außerdem sinnvoll Metadaten der Benutzer abzufragen, da so womöglich eine bessere statistische Auswertung möglich ist, damit zum Beispiel die Abhängigkeit bzw. die Unabhängingkeit der Stichproben erkennbar ist.

Insgesamt konnten die Zahlen zeigen, dass die generierten Tweets nicht den echten Donald Trump Tweets gleichwertig sind. Es hat sich aber gezeigt, dass der Trumpbot durchaus Tweets erzeugen kann, die denen von Donald Trump so stark ähneln, dass sie von Nutzern für echte Tweets gehalten werden.

References

[1] Wikipedia. https://en.wikipedia.org/wiki/Donald_Trump_on_social_media - Stand 31.08.2020.

[2] Trump Twitter Archive. http://www.trumptwitterarchive.com/ - Stand 31.08.2020.

[3] GloVe Twitter Embeddings. https://nlp.stanford.edu/projects/glove/ - Stand 31.08.2020.

[4] SkLearn Package. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html - Stand 31.08.2020.

[5] Markovify Package. https://github.com/jsvine/markovify - Stand 31.08.2020.

[6] GitHub Frontend Implementierung. https://github.com/vairasza/trumpbot_website - Stand 31.08.2020.

[7] GitHub Backend Implementierung. https://github.com/vairasza/trumpbot_lambda - Stand 31.08.2020.

[8] AWS Lamda. (https://aws.amazon.com/de/lambda/) - Stand 31.08.2020.