Twitter Network Mimicking for Data Storage Benchmarking

René Speck
Scalable Data Management
University Computing Centre
Leipzig University, Germany
Email: rene.speck@uni-leipzig.de

Diego Moussallem
DICE Group
Department of Computer Science
Paderborn University, Germany
Email: diego.moussallem@upb.de

Axel-Cyrille Ngonga Ngomo
DICE Group
Department of Computer Science
Paderborn University, Germany
Email: axel.ngonga@upb.de

***Abstract*—A significant portion of the textual data available on the Web comes from microblogging services such as Twitter. A considerable body of research has hence investigated methods for processing streams of short texts from such services as well as benchmarking online social networks. However, the costs connected with the acquisition of real microblogs is prohibitive for researchers with limited resources. We address this challenge by proposing TWIG, a benchmark generator for microblogging services similar to Twitter. It is a collection of algorithms to: 1) serialize users and their tweets from Twitter in RDF, 2) analyze such serialized data in RDF to approximate distributions over the underlying social network and 3) mimic a social network by generating synthetic users and tweets based on the approximated distributions. By using TWIG generated data, researchers can carry out preliminary evaluations of social network analysis and NLP approaches at low cost. Experimental and human evaluation results suggest that the synthetic tweets generated by TWIG are hardly distinguishable from human-generated tweets. Moreover, our results also underpin the scalability of our approach. Our Java implementation of TWIG is open-source and can be found at: https://github.com/dice-group/TWIG.**

I. Introduction

A significant portion of the textual data available on the Web comes from microblogging services. For example, the Statista service records approximate 330 million Twitter users in the first quarter of 2019.[1] A considerable body of research has hence investigated methods for processing streams of short texts from such services as well as benchmarking online social networks. The increasing popularity of microblogging services as a data source for a multitude of applications, such as entity extraction [1] and sentiment analysis [2], [3], makes the existence of corresponding benchmarks, which allow repeatable experiments, indispensable for the corresponding researchers. For this reason, a number of reference datasets based on Twitter (e.g., the Twitter7 dataset with 476 million tweets from 20 million users covering a six-month period from June 1 2009 to December 31 2009) were created [4]. However, a large portion of these datasets are no longer available for legal reasons. While corresponding APIs are available (e.g., the Twitter API[2]), the costs connected with the acquisition of human-generated microblog data from the corresponding APIs as well as the costs for their manual annotation with, for instance, using Amazon Mechanical Turk [3], is prohibitive for researchers with limited financial resources. We developed TWIG in an effort to alleviate this problem.

In this paper, we address the problem of generating Twitter-like data for benchmarking by providing the Twitter Benchmark Generator, TWIG. TWIG has two main goals: mimicking the Twitter Network datastream (including tweets and users) and storing the mimicked results in an RDF serialization. TWIG achieves these goals by learning from a crawl of Twitter containing millions of tweets and users.[3] The crawl includes unique IDs for users, user tweet times and the tweets themselves.

TWIG parses a crawl to generate a synthetic network with synthetic tweets. This generation is based on three probability distributions. One for the tweet daytime, which describes the probability that a tweet is sent out at a specific minute during a day. One probability distribution for the number of tweets, which describes the probability that a specific number of tweets is sent in a given time period. Finally, TWIG employs a Markov chain for the distributions of word predecessors and successors, which describes the probability that a word is being followed by another word under the assumption of a Markov chain. Those three TWIG probability distributions are used in an automation to mimic users, tweet times and to generate synthetic tweets.

The results of TWIG can be used, for instance, to benchmark storage systems w.r.t. their performance when faced with Twitter streams [5]. Moreover, TWIG can be used as a silver standard for various tasks including named entity recognition and sentiment analysis so as to cut down the costs linked with the creation of such benchmarks [3].[4] Using TWIG has the main advantage of leading to highly configurable and scalable experiments. Moreover, it is a deterministic algorithm and thus creates with the same parameter and data always the same synthetic social network with the same users and tweets. Hence, TWIG ensures repeatable experiments. By being integrated into the HOBBIT platform [6], TWIG ensure fully comparable and FAIR benchmarking.

[1] https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users

[2] https://developer.twitter.com/en/products/tweets

[3] The crawl cannot be made available for legal reasons.

[4] We are aware that the results of TWIG can only be used to create silver standard data and run preliminary experiments as they generated automatically.

The rest of this paper is structured as follows. After briefly describing recent approaches related to the areas present in the TWIG processes in Section II, we define the terminology and notation used in this paper in Section III. Subsequently, we introduce our proposed approach in Section IV and present our evaluation in Section V. Lastly, we conclude by discussing our results and future work in Section VI.

II. Related Work

A. Natural Language Generation

Natural Language Generation (NLG) is the process of generating coherent natural language text from non-linguistic data [7]. Although the community agrees on the text and speech output of these systems, there is far less consensus on what the input should be [8]. A wide range of inputs have been employed for NLG systems, including images [9], numeric data [10] and Semantic Web (SW) data [11]. Real applications can be found in domains such as health care [12], feedback for car drivers [13], diet management [14] and sportscasting news [15].

Recently, Twitter has made access on their data private, making only available its outdated NLP datasets such as the subsets for sentiment analysis[5]. Consequently, the research on noisy NLP text has been affected. A few works have investigate the automatic generation of tweets for training Natural Language Processing (NLP) models such as TwiBiNG by [16]. This work deals with the selection of relevant data on the enormous volumes of tweets generated every minute. They proposed a bipartite algorithm that clusters authentic tweets based on key phrases and ranks the clusters based on trends in each timeslot. Finally, we present an approach to select those topics which have sufficient content to form a story. Still, the researcher must have access on the twitter private key to run this research. [17] investigated the automatic generation of tweets, the authors created a data-driven NLG approach, which makes use of traffic information with other knowledge sources, to automatically generate natural language tweets. They considered how different forms of information can be combined to provide tweets customized to a particular location and/or specific user.

To the best of our knowledge, our approach is the first to create a benchmark generator for microblogging services similar to Twitter.

B. Text Classification

A plethora of works have exploited the problem of classifying short-text, mostly tweets. [18] employed Convolutional Neural Network (CNN) model augmented with a knowledge base to contextualize tweets for identifying the category of information contained in crisis-related social media content. [19] enriched the tweets' representation with word and subword information to be used on the same CNN architecture. Other works have also confirmed the efficiency of CNN in classifying tweets of even new events [20], [21]. Due to the promising results of end-to-end deep learning approaches, other Neural Network (NN) architectures gained traction on the same task. [22] trained a Bidirectional-LSTM as the first hidden layer to encode the contextual information, followed by three more layers of vanilla LSTM. Their experiments outperformed all previous CNN based models. Recently, [23] relied on BERT [24], which is the new state-of-the-art language model, to classify disaster-related tweets into multi-label information types. The authors employed a fine-tuned BERT model with ten BERT layers. In the same vein, [25] used BERT for transfer learning. The standard BERT architecture for classification and several other customized BERT architectures are trained to compare with the baseline bidirectional LSTM along with pre-trained Glove Twitter embeddings.

Some non-deep learning algorithms are effective and perform well. [26] explores a simple and efficient baseline for text classification. Our experiments show that the fastText classifier is often on par with deep learning classifiers.

III. Preliminaries and Notations

A. TWIG Ontology

Our TWIG Ontology, partly visualized with WebVOWL [27] in Figure 1, consists of three classes: `Tweet`, `OnlineTwitterAccount` and `SocialNetwork`. The last two mentioned are subclasses of classes from the external FOAF Vocabulary[6]. Furthermore, the TWIG Ontology consists of two object properties, `sends` and `mentions`, which describe sending tweets from a user within a social network and mentions of users in a tweet. Through this paper, we use the term user as a synonym for an instance of the `OnlineTwitterAccount` class. Each tweet has two data properties: `tweetContent`, a string holding the tweet message; and `tweetTime`, a dateTime holding the time a tweet was sent in the network.

The TWIG Ontology, which is prefixed with `twig` in this paper, can be found at the public repository[7].

B. RDF and SPARQL

Throughout the paper, it is assumed that the prefixes in Listing 1 are preposed whenever a Resource Description Framework (RDF) serialization in Turtle is listed in the paper and the prefixes in Listing 2 whenever a SPARQL Protocol and RDF Query Language (SPARQL) query is listed in the paper.

Listing 1: Prefixes for RDF serializations.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix twig: <https://dice-research.org/twig#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

[5]https://www.kaggle.com/kavita5/twitter-dataset-avengersendgame

[6]http://xmlns.com/foaf/0.1/

[7]https://github.com/dice-group/TWIG/blob/master/src/main/resources/OWL/twig.owl

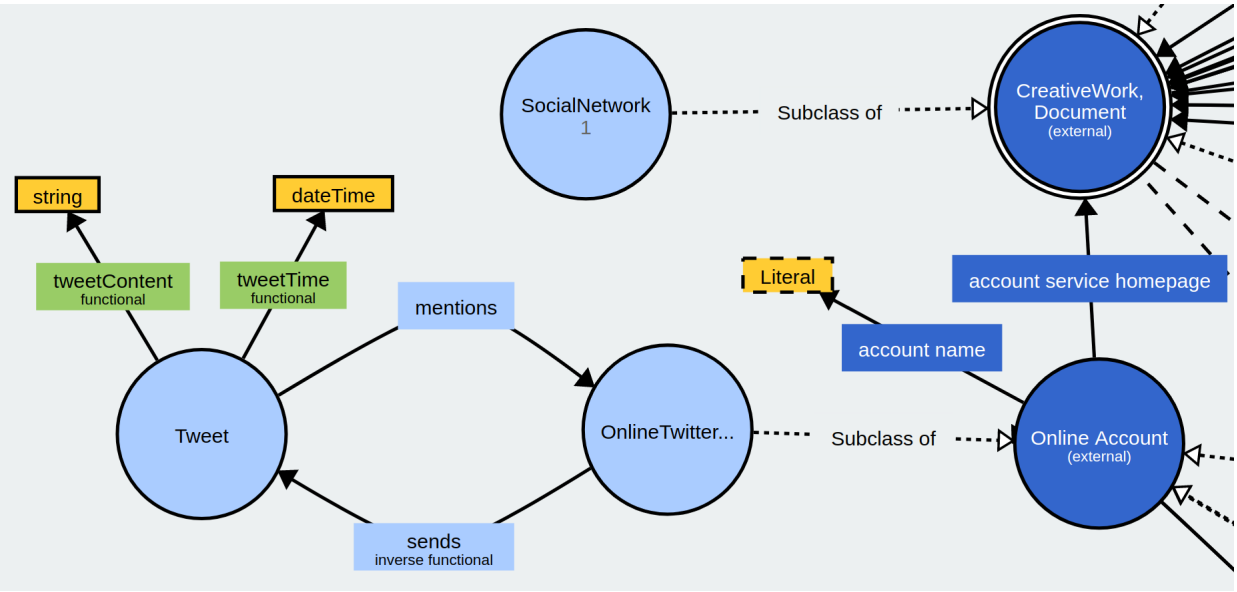

Figure 1: TWIG Ontology partly visualized.

Listing 2: Prefixes for SPARQL queries.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/> .
PREFIX owl: <http://www.w3.org/2002/07/owl#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX twig: <https://dice-research.org/twig#> .
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> .
```

C. Markov Chain Model

Markov chains are one of the stochastic processes that are widely applied in discrete time series modeling and simulation. We follow the definitions in [28], a Markov chain is defined as a special type of stochastic process, which deals with characterization of sequences of random variables, with particular attention paid to the dynamic and the limiting behaviors of the sequences. A stochastic process can be defined as a set of random quantities $\{\theta^{(t)} : t \in T\}$ from a set T. The set $\{\theta^t : t \in T\}$ is said to be a stochastic process with state space S and parameter set T. In this paper the set T is taken as countable and thus defining a discrete time stochastic process. A Markov chain is a stochastic process where given the present state, past and future states are independent

$$Pr(\theta^{(n+1)} \in A | \theta^{(n)} = x, \theta^{(n-1)} \in A_{n-1}, \ldots, \theta^{(0)} \in A_0) = Pr(\theta^{(n+1)} \in A | \theta^{(n)} = x)$$

for all sets $A_0, ..., A_{n-1}, A \subset S$ and $x \in S$. In this paper, $S = \{x_1, ..., x_r\}$ is a discrete state space and finite with r elements. In our approach, we consider a homogeneous, two-state, first-order Markov chain for which the probability of any particular transition from one step to the next remains constant over time and for which only two states exists, the predecessor and successor state. Thus, a transition matrix P, given by transition probabilities $P(x_i, x_j)$ on the (i, j)th elements can be defined

$$P = \begin{pmatrix} P(x_1, x_1) & \ldots & P(x_1, x_r) \\ \vdots & \ddots & \vdots \\ P(x_r, x_1) & \ldots & P(x_r, x_r) \end{pmatrix}.$$

IV. TWIG Approach

Our approach, TWIG, aims to mimic a given social network with it's users and blog messages. Therefore, it is in general a collection of algorithms to: 1) crawl and parse social network data (with users, their blog messages and sending times) as well as to anonymize, transform and serialize the parsed data into an RDF TWIG model; 2) analyze RDF TWIG models and approximate probability distributions of the underlying social network; 3) mimic social networks by generating users and microblogging messages based on the approximated distributions.

Figure 2 depicts the TWIG data flow. The dashed rectangles show the training (on the left) and the mimicking phase (on the right). Starting with the training phase, which includes our algorithms for crawling, parsing and analyzing, TWIG offers an ontology (Section III-A) that describes social networks with users and microblogging messages. Based on this ontology, a parser transforms a crawled or given dataset of a social network into an RDF TWIG model. This model stores the given authentic network in machine-readable form. Based on this, the authentic network is analyzed through querying this stored RDF TWIG model with SPARQL requests. With those analyses, three distributions over the authentic network are approximated and serialized. In the mimicking phase, those three distributions along with its parameters serve as input to mimic a social network. This synthetic social network is then serialized in RDF based on the TWIG ontology.

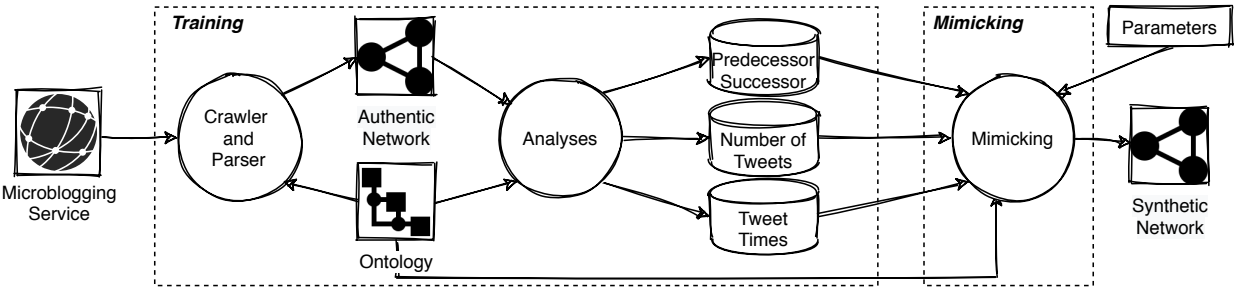

Figure 2: The data flow of the proposed approach.

A. TWIG Model

With TWIG, we crawl and parse a given microblogging service, anonymize the parsed data and store the data that include unique IDs for users, user tweet times and the tweets themselves into an RDF serialization. The TWIG parsing algorithm includes an anonymization step enabling its usage for other datasets with private data.

Listing 3: General stored user data.

```
twig:u
  a owl:NamedIndividual, twig:OnlineTwitterAccount ;
  twig:sends twig:u_t1, twig:u_t2 .
```

Listing 4: General stored tweet data.

```
twig:u_t1
  a owl:NamedIndividual, twig:Tweet ;
  twig:tweetContent m^^xsd:string ;
  twig:tweetTime t^^xsd:dateTime ;
  twig:mentions twig:v .
```

Let `u` be an anonymized user name. Further, let `u_t1` and `u_t2` two tweets sent by this user. TWIG stores a user and its sent tweets as an RDF resource (Listing 3); and a tweet as an RDF resource as well (Listing 4). A tweet sent by user `u` is stored as `u_t` where `t` is the tweets timestamp in the format `yyyy-mm-ddThh:mm:ss`. Let `m` be a tweet message of the tweet `u_t`, every matching of the regular expression `@([a-zA-Z0-9_]{1,15})` in `m` is a reference to another user `v`. For every reference to another user, TWIG will add those as mentions to the tweet `u_t`.

The result of the parsing process is a single RDF TWIG model containing all users and their tweets of the authentic network. We serialize this authentic network and the mimicked synthetic network by TWIG with the same schema. Thus, we basically utilize the TWIG ontology to describe the input and output data of our approach.

B. Analysis

Mimicking a network requires knowledge of the behavior of this network. Hence, TWIG offers analyses of given RDF twitter models, which are based on the ontology as it has been descried in Section IV-A. The analyses cover three different distributions: the Tweet Number Distribution ν, the Tweet Daytime Distribution δ and the Word Predecessor-Successor Distribution ω, which are discrete probability distributions.

1) Tweet Number Distribution: TWIG mimics the behavior of the number of tweets, send within a specified time t by a network user, with an approximated discrete probability distribution ν_t sampled on the original network. For this approximation, TWIG analysis the original network by querying over all users in the RDF model to count the number of tweets that have been sent by the single users (see Listing 5) and to record the difference between the first and the last day the user has sent at least one message (see Listing 6).

Listing 5: SPARQL query to request the number of tweets of a user u.

```
SELECT (COUNT(?tweet) AS ?n) WHERE {
  u twig:sends ?tweet
}
```

Listing 6: SPARQL query to request the first and last time a tweet was sent by user u.

```
SELECT (MIN(?tweetTime) as ?first)
       (MAX(?tweetTime) as ?last) WHERE {
  u twig:sends ?tweet .
  ?tweet twig:tweetTime ?tweetTime
}
```

Let $n \in \mathbb{N}$ be the number of tweets sent by a user u. Let $t, t' \in \mathbb{N}^+$, with t a constant number of days for the normalization of days and with t' the number of days a user u has sent messages. The normalized number of tweets is calculated as $n_t = \frac{n}{t'} \cdot t$. With this normalized number of tweets, a frequency distribution $H_{\nu,t} : \mathbb{N} \to \mathbb{N}$ is determined, where $H_{\nu,t}(j) = k$ expresses that k users have send j tweets in t days. Based on $H_{\nu,t}$, a discrete probability distribution $\nu_t : \mathbb{N} \to [0,1]$ can be determined. This probability distribution gives the probability for a user sending j tweets during t days: $\nu_t(j) = \frac{H_{\nu,t}(j)}{\sum_{i \in \mathbb{N}^+} H_{\nu,t}(i)}$ with $j \in \mathbb{N}$.

2) Tweet Daytime Distribution: TWIG mimics the behavior of the network daytime with an approximated probability distribution δ sampled on the original network. TWIG determines the frequency distribution $H_\delta : \Gamma \to \mathbb{N}$, where $\Gamma = \{00{:}01, 00{:}02, \ldots, 23{:}59\}$ is the set of minutes on a single day. Based on the frequencies gathered from the original network data (Listing 7), the probability distribution $\delta : \Gamma \to [0,1]$ is calculated

with: $\delta(j) = \frac{H_\delta(j)}{\sum_{\gamma \in \Gamma} H_\delta(\gamma)}$, which expresses on which minute during the day a tweet is send by a network user.

Listing 7: SPARQL query to request the time a tweet u_t1 was sent.

```
SELECT ?tweetTime WHERE {
  u_t1 twig:tweetTime ?tweetTime
}
```

Listing 8: SPARQL query to request the tweet content of tweet u_t1.

```
SELECT ?content WHERE {
  u_t1 twig:tweetContent ?content
}
```

3) Word Predecessor-Successor Distribution: TWIG makes use of a first-order Markov chain, which describes the probability for a word being followed by another word, to create synthetic tweets. TWIG queries (Listing 8) and analysis tweets in a given RDF TWIG model to create frequency distributions over all words and the followed words in the tweets. It applies a fast tokenizer [29] to split those messages to words. A set of characters Σ is given and a word predecessor-successor combination, a tuple of words (w_1, w_2) with $w_1, w_2 \in \Sigma^*$, defines that word w_2 is successor of word w_1. TWIG approximates a Word Predecessor-Successor Distribution $\omega : \Sigma^* \to \widehat{\omega}$ with $\widehat{\omega} : \Sigma^* \to [0, 1]$. Based on the frequencies of predecessor-successor combinations, $H_\omega : \Sigma^* \times \Sigma^* \to \mathbb{N}$ gathered from the tweet content, we created the probability distribution ω. Thus, with $(\omega(w_1))(w_2) = p$, the transition probability is p for (w_1, w_2).

C. Synthetic Tweet Generation and Network Simulation

TWIG applies the three distributions (Section IV-B) to mimic the Twitter network by creating synthetic tweets, users and there tweet times. TWIG expects parameters for the scalability of the data: parameter n for the number of users; d defines the start date; t the duration of the mimicking time in days and s a pseudo random number generator seed. The last parameter is needed for variations of the mimicking approach since TWIG is a deterministic algorithm and thus creates with the same parameter and data always the same users and synthetic tweets.

TWIG starts by initializing the machines pseudorandom number generator with the seed s and by creating n random user names. For each user name, it estimates the number of tweets a user sends per day with the help of the Tweet Number Distribution ν. For each of these tweets, TWIG creates the time when the tweet is send with the help of the Tweet Daytime Distribution δ. TWIG then creates the tweet content with the help of the Word Predecessor-Successor Distribution ω. The created synthetic network with users and tweets, which belong to the synthetic users, are serialized in an RDF TWIG model as descried in Section IV-A.

Listing 9: A snippet of a mimicked synthetic network by TWIG.

```
twig:65abfcdb2ed2939
  a owl:NamedIndividual, twig:OnlineTwitterAccount ;
  twig:sends twig:65abfcdb2ed2939_2009-10-01T00-28-23_1 ,
             twig:65abfcdb2ed2939_2009-10-01T00-02-12_1 .

twig:65abfcdb2ed2939_2009-10-01T00-28-23_1
  a owl:NamedIndividual , twig:Tweet ;
  twig:tweetContent "Get up! I asked me." ;
  twig:tweetTime "2009-10-01T00:28:23"^^xsd:dateTime.

twig:65abfcdb2ed2939_2009-10-01T00-02-12_1
  a owl:NamedIndividual , twig:Tweet ;
  twig:mentions twig:2ad82edc330b4f2 ;
  twig:tweetContent "I won't go home? \
                     Sounds like a chance to get your \
                     own self! @2ad82edc330b4f2" ;
  twig:tweetTime "2009-10-01T00:02:12"^^xsd:dateTime .
```

An snippet of a synthetic network mimicked by TWIG is given in Listing 9. This snippet consists of a user who has sent two tweets, one tweet with a mention of another user.

V. Experiments

We focus on the Twitter social network as an instance in our experiments. The aim of our experiments was to address the following questions regarding the short text generation with the trained TWIG models:

- Q_1 *Does TWIG produce synthetic tweets competitive to the quality of authentic tweets?*
- Q_2 *How does the training size of the text generation model influence the quality of the synthetic tweets?*

This paper experiments follow a two fold evaluation, a) an quantitative evaluation and b) a qualitative evaluation, to answer these two questions. All experiment data and results, which are non personal data, are public available at: https://github.com/dice-group/TWIG.

For the quantitative evaluation, we applied supervised text classifications to classify sets of tweets that comprised of authentic tweets from the Twitter network and synthetic tweets from TWIG. The text classifications aim was to distinguish between two classes of tweets, authentic and synthetic tweets. In the first step, we trained models of text classification algorithms, that we then used in the second step for classifications between authentic and synthetic tweets. To answer the influence of the training size of the TWIG model in the second question, we separated the quantitative evaluation in two scenarios. In the first scenario, models were trained on a small dataset, so that a trained model fits in the memory of a common home computer system. In the second scenario, models were trained on a much bigger dataset, thus the model size is not suitable for common home systems.

For the qualitative evaluation, synthetic tweets generated by TWIG and authentic tweets from the Twitter network were randomly chosen. Those tweets were randomized and then present to experts[8] with the task to classify those tweets in authentic and synthetic tweets.

[8]Humans with a degree in computer science.

A. Experimental Setup

1) Datasets: Twitter7 [30] is a multilingual dataset of tweets crawled from the Twitter network. Those tweets are of 17,069,982 users tweeted within a seven month period from June 1 2009 to December 31 2009. Thus, in average around 28 tweets per user. From the dataset, we deleted empty tweets and thus 476 323 635 were left from the Twitter7 dataset.

2) TWIG Model Training: Experiments were performed with two different TWIG model sizes. Since in our approach, the model size depends on the training data, we trained one TWIG model $\mathcal{S}$ with a small portion and another TWIG model $\mathcal{L}$ with a large portion of the Twitter7 dataset. The training data for model $\mathcal{L}$ consists of around 60% of the Twitter7 dataset,[9] whereas the training data for model $\mathcal{S}$ consists of around 4% of the Twitter7 dataset[10]. We trained the small model $\mathcal{S}$ on a common home computer system with four cores (Core i7) and with 16GB of RAM. The large model $\mathcal{L}$, was trained on 64 cores with 256GB of RAM.

3) Short Text Generation and Post-processing: Regarding the short text generation, TWIG implements just a simple approach to produce synthetic tweets by concatenating the words revived by the Word Predecessor-Successor Distribution with spaces. In our experiments, we observed that the concatenation of words to generated synthetic tweets has clearly influenced the quality of the synthetic tweets. Thus, we applied a post-processing step on the generated synthetic tweets to increase the quality by cleansing the tweets in this step. We discuss the results of this additional step in the result section. The following three post-processing steps were applied to synthetic tweets: We deleted all spaces in front of all commas, exclamation marks, opening brackets, points, question marks, semicolons; and we deleted all spaces after all closing brackets; and we removed all quotation marks, i.e., double quotations and inverted commas.

4) Evaluation with Supervised Text Classifications: Our quantitative evaluation comprised of two different algorithms, fastText [26] and BERT [24], to classify a set of tweets in authentic and synthetic. The first algorithm, fastText, assumes multi-label classification; thus, in our case, recall and precision values are equal. In our experiments with this algorithm, we ran each experiment three times and reported the averaged precision at one (P@1), based on [26]. For the second algorithm, BERT, we utilize the parameters for this algorithm as in [24] and we report the loss as well as the accuracy of our experiments.

Supervised text classification requires training of a prediction model. Thus, we created training datasets to train the text classification algorithms. For those training datasets, we chose 700k tweets from the Twitter7 dataset that were not chosen to train TWIG and 700k synthetic tweets generated by TWIG.

For the test dataset, we chose 300k tweets from the Twitter7 dataset that were also in the training dataset for TWIG and we chose 300k synthetic tweets generated by TWIG for the classification between authentic and synthetic. The test dataset were without retweets and user mentions, because Twitter7 comprises real usernames in tweets but tweets by TWIG are with anonymized usernames. Thus retweets and tweets with user mentions are very easy to classify in authentic or synthetic because of the different representations.

5) Evaluation with Experts: For the human evaluation, 50 tweets were randomly chosen from TWIG and 50 were randomly chosen from the Twitter network (tweets without retweets). Those 100 tweets were randomly present to humans who had the task to classify those tweets in authentic and synthetic tweets.

[9]More precisely 290 919 227 tweets.

[10]More precisely 18 568 442 tweets.

Table I: Precision on the two TWIG models and two methods with the fastText classification.

Methods	$\mathcal{S}$	$\mathcal{L}$
default	0.5696	**0.5666**
post-processing	0.5464	**0.5387**

Table II: Accuracy and loss on the two TWIG models with the BERT classification.

Measures	$\mathcal{S}$	$\mathcal{L}$
accuracy	**0.5680**	0.5874
loss	**4.0117**	4.2942

B. Results and Discussion

1) Quantitative Results: Tables I and II list our results observed with supervised text classifications on the small and the large trained TWIG model, $\mathcal{S}$ and $\mathcal{L}$.

Table I lists our evaluation results with the fastText text classification algorithm. Both models, $\mathcal{S}$ and $\mathcal{L}$, reach nearly the same performance with 0.57 P@1 without post-processing. With post-processing, the performance increase on both models, 4.25% on $\mathcal{S}$ and 5.18% on $\mathcal{L}$. The best performance in this scenario is marked in bold and the results suggest that the synthetic tweets are hard to distinguish from authentic tweets because the results are below 0.6 P@1.

Table II lists our evaluation results with the BERT text classification algorithm. In this scenario, we applied the post-processing because it increased the performance in our first experiments. The results suggest that the performance on the small model is slightly better than on the large model, with 0.57 to 0.59 accuracy. The smaller value of the loss function on S, also indicates the better performance. In this scenario, the accuracy is below 0.6 on both models. An accuracy of 0.5 means not distinguishable. Consequently, these experiments suggest that synthetic tweets are hardly distinguishable from authentic tweets.

It is worth noticing the work of [31] in terms of these results, it shows that BERT is not always a powerful language model and can be fooled by attackers via a simple adversarial strategy. An attacker does not even need to use grammatical or semantically meaningful queries to break BERT.

2) Qualitative Results: In our qualitative evaluation, 15 experts were involved and split into five groups of three experts in each group. Our evaluation dataset consists of 100 randomly sampled tweets, 50 authentic tweets from Twitter7, and 50 synthetic tweets produced by TWIG. We divided those 100 tweets into five parts with 20 tweets each. Each of the five parts were processed by one group of three experts. Thus, each expert in one group classified the same 20 tweets. The final classification of a group is the majority of the expert answers in each part, with 34 true positives, 15 false positives, 16 false negatives, and 35 true negatives. Hence TWIG achieves 0.69 accuracy in this experiment on our dataset with 100 tweets, with 32% of the synthetic tweets were wrongly classified as authentic tweets.

Five examples of synthetic messages that were wrongly classified as authentic tweets are listed in the following:

- Please take pressure is being a governor run its not easy!! lot but am being drunk so silly... :)
- He could win something meaningful and cool nutrition. :) LOL... i save a
- I'm movin #TYPD #TYPD
- Going to forex market.
- Right. Havent Been diving! #fb

3) Discussions: Our quantitative evaluation showed that some state-of-the-art text classification algorithms could hardly distinguish tweets whether they are authentic or synthetic. In addition, our qualitative evaluation showed that 32% of the synthetic tweets were wrongly classified as authentic tweets. These results lead to the answer to our first question:

Answer to Q_1: A significant portion of synthetic tweets were wrongly classified as authentic tweets by human experts. Consequently, TWIG produces synthetic tweets competitive with the quality of authentic tweets.

Also, in our quantitative evaluation, we investigated two TWIG models, which were trained on different dataset sizes, a small and a large one. The results suggest, that our approach with the model trained on a relatively small training dataset, which fits in a common home computer, achieves almost the same results compared to the model trained on a much larger dataset. These results lead to the answer to our second question:

Answer to Q_2: The results on the two different models suggest, that our approach reaches good results on the model trained with a relative small training data. Increasing the training data barely increases the performance of our approach.

VI. Conclusion and Future Work

We presented TWIG, a collection of algorithms to mimic social networks for benchmark generation. We showed how our approach approximates an underlying social network with three probability distributions for mimicking this network with its users, text messages and daytimes. Our results unveil a number of questions and suggest that TWIG's text generator can be improved further by improving the word and sentence semantics incorporated in generating text. Thus, in the near future, we plan to integrate semantic embedding in our text generator with the aim to improve the performance particularly by increasing the semantics in the generated synthetic tweets.

Further our results regarding the post-processing suggest, that a document and sentence planner could also improve the quality of the text generation within TWIG and thus, we plan to integrate those.

Acknowledgements

This work has been supported by the BMWI Project RAKI (Project No. 32100687, Funding No. 01MD19012D).

References

[1] U. K. Sikdar and B. Gambäck, "Twitter named entity extraction and linking using differential evolution," in *Proceedings of the 13th International Conference on Natural Language Processing*, 2016, pp. 198–207.

[2] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. J. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, 2011, pp. 30–38.

[3] D. Zimbra, A. Abbasi, D. Zeng, and H. Chen, "The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 2, pp. 1–29, 2018.

[4] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 177–186.

[5] M. Spasić and M. Jovanovik, "Mocha 2017 as a challenge for virtuoso," in *Semantic Web Evaluation Challenge*. Springer, 2017, pp. 21–32.

[6] M. Röder, D. Kuchelev, and A.-C. Ngonga Ngomo, "Hobbit: A platform for benchmarking big linked data," *Data Science*, no. Preprint, pp. 1–21, 2019.

[7] E. Reiter and R. Dale, *Building natural language generation systems*. Cambridge university press, 2000.

[8] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," *Journal of Artificial Intelligence Research*, vol. 61, pp. 65–170, 2018.

[9] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

[10] D. Gkatzia, H. F. Hastie, and O. Lemon, "Comparing multi-label classification with reinforcement learning for summarisation of time-series data." in *ACL (1)*, 2014, pp. 1231–1240.

[11] A.-C. Ngonga Ngomo, D. Moussallem, and L. Bühman, "A Holistic Natural Language Generation Framework for the Semantic Web," in *Proceedings of the International Conference Recent Advances in Natural Language Processing*. ACL (Association for Computational Linguistics), 2019, p. 8.

[12] F. Portet, E. Reiter, A. Gatt, J. Hunter, S. Sripada, Y. Freer, and C. Sykes, "Automatic generation of textual summaries from neonatal intensive care data," *Artificial Intelligence*, vol. 173, no. 7–8, pp. 789 – 816, 2009.

[13] D. Braun, E. Reiter, and A. Siddharthan, "Saferdrive: An NLG-based behaviour change support system for drivers," *Natural Language Engineering*, vol. 24, no. 4, pp. 551–588, 2018.

[14] L. Anselma and A. Mazzei, "Designing and testing the messages produced by a virtual dietitian," in *Proceedings of the 11th International Conference on Natural Language Generation*. Tilburg University, The Netherlands: Association for Computational Linguistics, Nov. 2018, pp. 244–253. [Online]. Available: https://www.aclweb.org/anthology/W18-6531

[15] C. van der Lee, E. Krahmer, and S. Wubben, "PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences," in *Proceedings of the 10th International Conference on Natural Language Generation*, ser. INLG'2017. Santiago de Compostela, Spain: Association for Computational Linguistics, 2017, pp. 95–104. [Online]. Available: http://aclweb.org/anthology/W17-3513

[16] Y. Sharma, D. Bhatia, and V. K. Choudhary, "Twibing: A bipartite news generator using twitter," in *Proceedings of the SNOW 2014 Data Challenge co-located with 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014*, ser. CEUR Workshop Proceedings, S. Papadopoulos, D. Corney, and L. M. Aiello, Eds., vol. 1150. CEUR-WS.org, 2014, pp. 70–76. [Online]. Available: http://ceur-ws.org/Vol-1150/sharma.pdf

[17] K. Tran and F. Popowich, "Automatic tweet generation from traffic incident data," in *Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web (WebNLG 2016)*, 2016, pp. 59–66.

[18] G. Burel, H. Saif, and H. Alani, "Semantic wide and deep learning for detecting crisis-information categories on social media," in *International Semantic Web Conference*. Springer, 2017, pp. 138–155.

[19] C. Caragea, A. Silvescu, and A. H. Tapia, "Identifying informative messages in disaster events using convolutional neural networks," in *International conference on information systems for crisis response and management*, 2016, pp. 137–147.

[20] J. Wang, Z. Wang, D. Zhang, and J. Yan, "Combining knowledge with deep convolutional neural networks for short text classification." in *IJCAI*, 2017, pp. 2915–2921.

[21] D. T. Nguyen, K. A. Al Mannai, S. Joty, H. Sajjad, M. Imran, and P. Mitra, "Robust classification of crisis-related data on social networks using convolutional neural networks," in *Eleventh International AAAI Conference on Web and Social Media*, 2017.

[22] S. Kundu, P. Srijith, and M. S. Desarkar, "Classification of short-texts generated during disasters: A deep neural network based approach," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 790–793.

[23] H. M. Zahera, I. A. Elgendy, R. Jalota, and M. A. Sherif, "Fine-tuned BERT model for multi-label tweets classification," in *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019*, ser. NIST Special Publication, E. M. Voorhees and A. Ellis, Eds., vol. 1250. National Institute of Standards and Technology (NIST), 2019. [Online]. Available: https://trec.nist.gov/pubs/trec28/papers/DICE_UPB.IS.pdf

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[25] G. Ma, "Tweets classification with bert in the field of disaster management," 2019.

[26] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 427–431. [Online]. Available: https://www.aclweb.org/anthology/E17-2068

[27] S. Lohmann, V. Link, E. Marbach, and S. Negru, "Webvowl: Web-based visualization of ontologies," in *Knowledge Engineering and Knowledge Management*, P. Lambrix, E. Hyvönen, E. Blomqvist, V. Presutti, G. Qi, U. Sattler, Y. Ding, and C. Ghidini, Eds. Cham: Springer International Publishing, 2015, pp. 154–158.

[28] D. Gamerman, *Markov chain Monte Carlo : stochastic simulation for Bayesian inference*, 2nd ed., ser. Texts in statistical science. Chapman & Hall, 2006.

[29] O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, and N. Smith, "Improved part-of-speech tagging for online conversational text with word clusters," *Proceedings of NAACL-HLT*, vol. 2013, pp. 380–390, 01 2013.

[30] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *WWW '10: Proceedings of the 19th international conference on World wide web*. New York, NY, USA: ACM, 2010, pp. 591–600.

[31] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer, "Thieves on sesame street! model extraction of bert-based apis," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=Byl5NREFDr