Entwicklung eines Tools zur automatisierten Suche nach Tweets zum Thema "Phishing"

Julia Lutzweiler

Institut für Angewandte Technisch-Kognitive Systeme
Karlsruher Institut für Technologie

Abstract. Um mehr Bewusstsein für Phishingangriffe im Internet zu schaffen, möchte die Forschungsgruppe SECUSO gezielt auf Twitter mit Informationsmaterial werben. Ziel dieser Arbeit ist es, Tweets zum Thema *Phishing* zu finden und so aufzubereiten, dass auf die relevantesten Tweets leicht geantwortet werden kann. Dazu werden mit einer Twitter-API zunächst Tweets, die den Suchbegriff enthalten, heruntergeladen. Anschließend werden sie durch ein Priorisierungsverfahren in eine sinnvolle Reihenfolge gebracht. Das Tool wurde so implementiert, dass es flexibel eingesetzt werden kann. Einerseits ist es möglich die Suchbegriffe für andere Anwendungszwecke anzupassen, andererseits ist es möglich, es in verschiedenen Betriebsmodi auszuführen. Dazu gehören eine Anbindung an ein Web-Frontend, automatische Übermittlung der Ergebnisse per E-Mail und direkte Ausführung im Terminal.

1 Einleitung

Mit *Phishing* bezeichnet man Versuche eines Angreifers im Internet durch Vortäuschen einer falschen Identität an private Informationen zu gelangen[1]. Damit Nutzer im Internet sich besser vor solchen Attacken schützen können, bietet die Forschungsgruppe SECUSO Informationsmaterialien[1] zum Thema Phishing an. Um besser auf dieses Angebot aufmerksam machen zu können, wird auf den sozialen Medien dafür geworben. Auf der Microbloggingplattform Twitter soll gezielt auf Nutzer geantwortet werden, die Phishing erwähnen.

Auf Twitter werden Kurznachrichten, sogenannte *Tweets* ausgetauscht. Diese werden den *Followern* des Verfassers angezeigt, sind aber in der Regel auch öffentlich einsehbar. Andere Nutzer können mit einem *Like* zeigen, dass ihnen ein Tweet gefällt oder durch einen *Retweet* an ihre eigenen Follower verbreiten. Außerdem ist es möglich auf Tweets zu antworten, wobei auch die Antworten selbst Tweets sind.

Aufgrund des relativ einfachen Konzepts und der überwiegend öffentlichen Funktionsweise von Twitter ist es möglich, Tweets zu durchsuchen und in einem eigenständigen Programm zu verarbeiten. Ziel dieses Projekts ist es, ein Tool zu entwickeln, dass die Suche nach Tweets zum Thema Phishing ermöglicht.

[1] `https://secuso.aifb.kit.edu/betruegerische_nachrichten_erkennen.php`

Dadurch soll es möglich sein einfacher und gezielter relevante Informationsmaterialien zu verbreiten.

Zunächst geben wir einen Überblick über die API, die zum Zugriff auf Twitter angeboten wird. Es folgt eine Analyse der gefundenen Tweets aus dem Zeitraum einer Woche, die als Grundlage weiterer Überlegungen dienen. Anschließend wird der Aufbau unseres Tools erläutert. Im Fokus stehen hier die Suche und Priorisierung als Kernfunktionalität, trotzdem beschreiben wir auch die Komponenten, die zur Erweiterung dienen.

2 Implementierung

2.1 Twitter API

Um relevante Tweets abzufragen wurde die von Twitter zur Verfügung gestellte v2-API[2] verwendet. Je nach *Access Level* gelten unterschiedliche Einschränkungen. Das niedrigste Level, *Essential Access* ist frei verfügbar, allerdings können pro Monat maximal 500.000 Tweets abgerufen werden. Außerdem sind nicht alle Endpoints der API verfügbar. Die geringsten Einschränkungen gelten für das *Academic Research Level* das für akademische Projekte beantragt werden kann.

Für die meisten Funktionalitäten von Twitter gibt es einen entsprechenden Endpoint der API. Dazu zählen das Lesen und Verfassen eines Tweets, das Einsehen von Nutzerprofilen sowie Interaktionen (Likes, Retweets) mit anderen Tweets. Für unsere Anwendung ist in erster Linie das Suchen nach Tweets mittels Stichworten interessant. Hierzu gibt es die folgenden Endpoints:

- `/2/tweets/search/recent`: Suche in den Tweets der letzten Woche
- `/2/tweets/search/all`: Suche in allen jemals gesendeten Tweets
- `/2/tweets/sample/stream`: Ein Stream von einem Prozent aller Tweets

Unter diesen Endpoints ist nur `search/recent` mit dem *Essential Access* verfügbar. Da mit den relevanten Tweets zeitnah interagiert werden soll ist die Suche in älteren Tweets nur von geringem Vorteil. Mit dem `sample/stream` Endpoint wäre es möglich Suchanfragen aus komplexeren Kriterien zu verarbeiten. Allerdings müsste die Suche dann vollständig auf der Clientseite durchgeführt werden, was aufgrund der großen Datenmenge mit enormem Rechenaufwand verbunden wäre. Außerdem würde nur ein Prozent der tatsächlich gesendeten Tweets zur Verfügung stehen. Daher ist dieser Endpoint nur für statistische Auswertungen sinnvoll, nicht aber für ein Programm, dass mit individuellen Tweets arbeiten soll. Im Gegensatz dazu findet die Suche mit den ersten beiden Endpoints auf der Seite von Twitter statt. Zu der monatlichen Begrenzung zählen dann nur die Suchtreffer, aber nicht die Menge der durchsuchten Tweets.

Wir haben uns für den Endpoint `/2/tweets/search/recent` entschieden, da er für unsere Zwecke ausreichend und glücklicherweise auch direkt verfügbar ist.

[2] `https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api`

Zur Anbindung der API an unser Programm wurde das Python-Paket *tweepy*[3] benutzt. Es bietet Bindings zu allen Endpoints der Twitterschnittstelle und ist leicht zu verwenden.

2.2 Analyse der Tweets

Um einen Überblick über die Tweets zum Thema Phishing zu gewinnen, haben wir zunächst alle Tweets einer Woche analysiert. Diese Tweets wurden mit dem `search/recent`-Endpoint und der Suchanfrage `phishing lang:de` heruntergeladen. Anschließend konnten wir vier grobe Kategorien von Tweets finden. Sie unterscheiden sich durch das Nutzerprofil des Verfasser, dem Inhalt des Textes sowie der Relevanz für unser Projekt.

Konversationen Hier handelt es sich um Gespräche zwischen Personen die ihren Twitteraccount selbst nutzen, also nicht Accounts einer Institution wie ein Newsmagazin oder einer Forschungseinrichtung. Die Personen berichten in der Regel, wie sie selbst von Phishing betroffen waren. Sie könnten also von Informationen zum Thema profitieren und sind daher für uns direkt relevant. Entsprechende Tweets sind durch Klarnamen (zumindest der Vorname) im Profil und umgangssprachliche Formulierungen erkennbar.

News Hier berichten News- und Fachmagazine über ihren Twitteraccount über aktuelle Phishingversuche oder machen generell auf das Thema aufmerksam. Die Tweets sind am Nutzernamen direkt erkennbar und in der Regel auch durch Twitter offiziell verifiziert. Der Text enthält oft einen weiterführenden Link. Diese Tweets sind insofern für uns relevant, dass sie von vielen Menschen gesehen werden. Obwohl das Magazin die entsprechenden Informationen selbst nicht benötigt, können trotzdem viele Menschen erreicht werden.

Werbung Andere Einrichtungen werben für ihre Produkte oder eigenes Informationsmaterial die gegen Phishing helfen sollen. Es macht also wenig Sinn hier für eigene Informationen zu werben.

Spam Hier handelt es sich um irrelevante Tweets die hauptsächlich aus kryptischen Links und Hashtags bestehen. Oft enthalten diese Tweets so wenig Text, dass sie fälschlicherweise als deutsch identifiziert werden oder grundsätzlich keiner Sprache direkt zugeordnet werden können. Die Tweets werden von Accounts gesendet, die wenige Follower haben, dafür aber sehr viele Tweets verfassen. Außerdem sind viele Tweets in dieser Kategorie von nur wenigen Accounts erstellt worden.

Insgesamt wurden alle 809 Tweets aus dem Suchergebnis kategorisiert. Es handelt sich um Tweets, die zwischen dem 01.05.2022 und 08.05.2022 verfasst wurden. Die Tweets sind folgendermaßen aufgeteilt:

[3] `https://docs.tweepy.org/en/stable/index.html`

Kategorie	Anzahl	Anteil
Konversationen	104	13%
News	125	15%
Werbung	93	11%
Spam	487	60%
Insgesamt	809	100%

Die Kategorien *Konversationen* und *News* sind für uns am relevantesten. Da die Kategorisierung von Hand durchgeführt wurde und die Kategorien nicht präzise definiert sind, ist es nicht direkt möglich ein entsprechendes Programm zu schreiben das die Kategorisierung durchführt. Stattdessen haben wir uns entschieden, relevante Tweets in zwei Schritten zu bestimmen: Zuerst sollen alle potentiell interessanten Tweets gefunden werden. Dies geschieht durch die Suchfunktion auf der Seite von Twitter. Anschließend folgt in einem zweiten Schritt die Priorisierung der Tweets durch eine statistische Heuristik. Wenn die Priorisierung gut genug ist, müssen Tweets die für das Projekt nicht von Interesse sind nicht herausgefiltert werden, da sie sowieso am Ende der Liste erscheinen.

Bisher haben wir nur Ergebnisse zu dem Suchbegriff "Phishing" betrachtet. Es ist aber möglich, dass relevante Tweets den Suchbegriff nicht enthalten. Einerseits kann der Begriff in anderen grammatikalischen Formen auftreten oder falsch geschrieben sein. Wenn man zusätzlich die Suchterme "gefisht", "gefished" oder "phishen" mit einschließt, erhält man nur wenig mehr Ergebnisse. Erweitert man den Suchraum zusätzlich um andere Begriffe, die vielleicht von manchen Personen nicht differenziert werden, wie zum Beispiel "hacken", ist das Suchergebnis so groß, dass die eigentlich gewünschten Tweets nur einen kleinen Anteil ausmachen. Für das Beispiel "hacken" und grammatikalische verwandte Formen findet man in einer Woche über 100.000 Tweets, also weitaus mehr also für "phishing".

Andererseits kann das Thema aber auch ohne die Verwendung des Begriffs umschrieben werden. Das ist insbesondere problematisch, da wir solche Tweets mit unserer Suche nicht finden können und es wahrscheinlich gerade die Nutzer betrifft, die am meisten von Informationen zu Phishing profitieren könnten. Hier würde eine grundlegend andere Herangehensweise benötigt werden, etwa mit dem `sample/stream`-Endpoint und einer semantischen Analyse der Tweets.

2.3 Kernkomponenten

Unser Tool besteht aus zwei Kernkomponenten, die nacheinander ausgeführt werden. Zuerst wird eine Suchanfrage an Twitter gestellt und die Ergebnisse in einer Liste von `Tweet`-Objekten gespeichert. Anschließend wird für jeden Tweet ein Prioritätswert berechnet, der dann zur Sortierung der Liste verwendet wird. In diesem Abschnitt werden diese beiden Komponenten näher erläutert.

Suchanfrage Bevor eine Anfrage an Twitter gestellt wird, muss zunächst der Suchterm erzeugt werden. Da wir nach mehr als nur einem Begriff suchen wollen

können wir die einzelnen Begriffe mit `OR` verknüpfen um weniger Anfragen stellen zu müssen. Für die Suche spielt Groß- und Kleinschreibung keine Rolle.

Außerdem können verschiedene Operatoren hinzugefügt werden, die das Suchergebnis weiter einschränken. So kann man beispielsweise direkt nach Hashtags suchen. Allerdings ist anzumerken, dass die Suche nach einem Begriff auch Hashtags mit einschließt, das heißt dass bei der Suche nach "Phishing" auch der Hashtag "#Phishing" gefunden wird. Wir schränken mit dem `lang:`-Operator die Suche auf eine gewünschte Sprache wie sie Twitter klassifiziert ein. Obwohl das Ziel des Tools ist nach Phishing zu suchen, sind die Suchterme und die Sprache in einer Konfigurationsdatei wählbar. Mit den Standardeinstellungen wird die Anfrage `phishing OR phishen OR gephisht lang:de` erzeugt.

Anschließend wird die die Suchanfrage über den `search/recent`-Endpoint gestellt. Dabei kann angegeben werden, welche Daten zu den Tweets in der Antwort enthalten sein sollen. Das Ergebnis wird dann seitenweise zurückgegeben. Eine Seite enthält maximal 100 Tweets, wenn es mehr Suchergebnisse gibt, wird ein `next`-Token mitgeschickt, mit welchem die nächste Seite angefordert werden kann.

Wir iterieren über alle Suchergebnisse und konvertieren sie in eigene Tweetobjekte, die für unsere Zwecke nützlicher sind als die Tweetobjekte von *tweepy*. Für jeden Tweet speichern wir die folgenden Daten:

id	Jeder Tweet hat eine von Twitter vergebene ID. Sie wird benötigt um später auf einen Tweet antworten zu können.
time	Zeit und Datum als der Tweet verfasst wurde.
text	Der eigentliche Inhalt des Tweets.

Weiter speichern wir Daten bezüglich des Verfassers:

at_name	Das eindeutige mit dem @-Symbol beginnende Handle des Nutzers.
display_name	Der Name des Nutzers, wie er auf Twitter hervorgehoben angezeigt wird.
num_followers	Anzahl der Follower des Nutzers.
is_verified	gibt an, ob der Nutzer durch Twitter verifiziert wurde.

Zudem gibt es für jeden Tweet Metriken die für die Priorisierung von Bedeutung sind:

num_likes	Wie viele Nutzer den Tweet geliked haben.
num_retweets	Wie viele Nutzer den Tweet retweeted, also an ihre eigenen Follower verbreitet haben.
num_responses	Anzahl der Antworten auf den Tweet.
num_quotes	Ähnlich zu Retweets, allerdings mit einem zusätzlichen Kommentar versehen.

Letztenendes fügen wir noch zwei zusätzliche Felder hinzu, die erst an späterer Stelle gesetzt werden:

priority	Der noch zu berechnende Prioritätswert dieses Tweets.
done	Hiermit kann der Tweet als abgearbeitet vermerkt werden.

Die Suchergebnisse werden von Twitter in chronologischer Reihenfolge zurückgegeben. Durch einen optionalen Parameter in unserer Suchmethode erlauben wir es, den Vorgang vorzeitig abzubrechen, wenn der Tweet älter als ein gewisser Schwellenwert ist. Dadurch ist es möglich zu einer bereits vorhandenen Liste von Tweets nur neuere Tweets abzufragen und die Liste damit zu aktualisieren.

Priorisierung Anschließend wollen wir die Liste der Tweets priorisieren. Das geschieht indem jedem Tweet ein Prioritätswert P zugewiesen wird, der dann als Sortierkriterium dient. Im Programmcode wurde das mittels einer Prioritätsfunktion, die einen Tweet auf den Prioritätswert abbildet umgesetzt. Dadurch ist es möglich eine Prioritätsfunktion durch eine andere auszutauschen ohne Änderungen am Programmfluss vornehmen zu müssen.

Konkret berechnet unsere Prioritätsfunktion den Prioritätswert als ein gewichtetes Mittel der statistischen Eigenschaften eines Tweets. Dazu seien x_i die Größe eines Kriteriums, zum Beispiel die Anzahl der Likes, und w_i das entsprechende Gewicht. Dann berechnet sich der Prioritätswert P aus n Eigenschaften folgendermaßen:

$$P = \sum_{i=1}^{n} w_i \cdot x_i$$

Strikterweise müsste für ein gewichtetes Mittel die Summe der Gewichte 1 sein oder ein Normalisierungsfaktor hinzugefügt werden. Da wir nur an den relativen Unterschieden interessiert sind können wir darauf aber verzichten.

Für die Priorisierung verwenden wir die folgenden Kriterien: Die Anzahl der Follower des Accounts, sowie der Verifikationsstatus, die Zeit die seit dem Veröffentlichen des Tweets vergangen ist sowie die Anzahl der Likes, Retweets, Responses und Quotes. Um sinnvolle Ergebnisse zu erzielen müssen wir die Daten allerdings in einen gemeinsamen Zahlenbereich abbilden, denn die Zeit in Stunden, ein Wahrheitswert für den Verifikationsstatus und Metriken die potentiell über mehrere Größenordnungen gestreut sind, sind nicht direkt miteinander vergleichbar.

Für den Verifikationsstatus setzen wir $x = 1$ falls der Nutzer verfiziert ist und $x = 0$ falls nicht. Um den weit gestreuten Wertebereich der Metriken zu verkleinern verwenden wir den Logarithmus. Dadurch ist x selbst für Werte im Millionenbereich kleiner als 10. Allerdings muss beachtet werden, dass $\log(0)$ undefiniert ist, aber ein Tweet ohne Likes regelmäßig vorkommt. Sei y_i die Anzahl der Follower des Nutzers oder eine Metrik des Tweets. Dann berechnen wir die Werte x_i für die gewichtete Summe durch

$$x_i = \log(y_i + 1)$$

Das Alter des Tweets belassen wir in Stunden und gleichen den verhältnismäßig großen Zahlenbereich durch ein kleines Gewicht aus. Ein weiterer Unterschied ist das Vorzeichen des Gewichts. Ein großer Wert für P bedeutet, dass uns

der entsprechende Tweet wichtig ist. Alte Tweets verlieren an Relevanz, daher ist das Gewicht für das Alter negativ. Alle anderen Kriterien haben ein positives Gewicht, da sie darauf hindeuten, dass der Tweet von mehr Nutzern gesehen wird, oder dass die Person vertrauenswürdiger ist.

Um die Priorisierung auch anwenden zu können müssen noch angemessene Gewichte gewählt werden. Wir haben dem Gewicht für die Anzahl der Likes den Wert 1 gegeben und die anderen Gewichte im Verhältnis dazu gewählt:

Kriterium	Gewicht
Alter	−0.02
Verifikationsstatus	5
Anzahl der Follower	1
Anzahl der Likes	1
Anzahl der Retweets	5
Anzahl der Responses	2
Anzahl der Quotes	2

2.4 Erweiternde Funktionalität

Um das Tool auch effektiv einsetzen zu können, wurde es um weitere Funktionalität ergänzt. Im folgenden werden die entsprechenden Features kurz erläutert.

CSV-Export Um auf die Tweets, die eine hohe Priorität bekommen haben, antworten zu können, muss die Ausgabe des Programms in ein flexibles Format gebracht werden. Da es sich bei den Tweets um Daten handelt, die gut in einer Zeile Text darstellbar sind, eignet sich das `.csv`-Format hier gut. Nach der Priorisierung werden alle Tweets nach absteigender Priorität sortiert in eine `.csv`-Datei geschrieben. Diese Datei kann dann von vielen Programmen, beispielsweise Microsoft Excel, dargestellt werden. Hieraus lassen sich dann die Tweets im Browser über einen Link öffnen und können nach dem Abarbeiten abgehakt werden.

Außerdem dient die Datei auch als Speicher zwischen den Programmausführungen. Beim nächsten Programmstart wird die Datei wieder eingelesen, wodurch weniger neue Tweets heruntergeladen werden müssen. Dadurch werden die Tweets aus der `.csv`-Datei nicht überschrieben wodurch Tweets, die bereits als bearbeitet markiert wurden, nicht auf unbearbeitet zurückgesetzt werden.

Automatischer E-Mailversand Das Tool kann auch im Hintergrund regelmäßig ausgeführt werden. In diesem Fall muss der Nutzer von den aktuellen Ergebnissen benachrichtigt werden. Dazu wurde das Programm um ein Modul ergänzt, das nach dem zuvor beschriebenen `.csv`-Export automatisch eine E-Mail versendet. Als Anhang wird die exportierte Datei mitgeschickt.

Zur Umsetzung wurde das Pythonpaket *smtplib*[4] verwendet. Hiermit ist es möglich mit einem SMTP-Server Verbindung aufzunehmen und von dort eine E-Mail zu verschicken. Obwohl es theoretisch möglich ist, sich mit einem beliebigen SMTP-Server zu verbinden, ist nicht immer eindeutig, welche Konfiguration zum Verbindungsaufbau gefordert wird. Die Methode zum versenden einer E-Mail ist zur Nutzung des KIT-Smarthost-Servers ausgelegt. Dieser ist für das Senden von E-Mails, die von einem Gerät oder Dienst ausgehen, konzipiert.

Anschluss an Flask Anstelle wie oben beschrieben die Ergebnisse von einer exportierten Datei aus zu verwalten, soll es auch möglich sein, dies über ein Webinterface zu tun. Das erstellen des Web-Frontends war nicht Teil des Projektes, allerdings sollte das entsprechende Backend implementiert werden.

Für das Backend wurde das Python-Framework *Flask* verwendet. Es ermöglicht es, auf HTTPS-Anfragen mittels annotierter Funktionen zu antworten. Die verschiedenen Anfragen werden als *Routes* bezeichnet. Es wurden vier solcher Routes implementiert:

/tweets/ Gibt eine Liste aller Tweet-IDs als JSON-Array zurück. Die Tweet-IDs werden als String kodiert, da sie die maximale Größe eines JSON-Integers überschreiten.

/tweets/<id> Ruft einen Tweet über eine ID ab. Es wird ein JSON-Dictionary übergeben, dass alle Daten der `Tweet`-Klasse enthält.

/tweets/<id>/done Markiert den Tweet der entsprechenden ID als bearbeitet.

/tweets/<id>/undone Markiert den Tweet der entsprechenden ID als unbearbeitet.

Wenn das Programm mit Flask gestartet wird, wird zuerst die Liste der Tweets aktualisiert, danach werden die Flask-Routes erstellt und der Server wartet auf Anfragen. Die Daten werden trotzdem in der `.csv`-Datei gespeichert, um Kompatibilität mit den anderen Anwendungsmöglichkeiten zu gewährleisten. Außerdem gehen so die Daten beim Neustart des Servers nicht verloren. Aktuell ist es nicht möglich, neue Tweets während der Laufzeit des Servers abzufragen. Eine entsprechende Änderung sollte aber leicht umsetzbar sein.

Konfiguration Da das Tool mit einigen anderen Diensten interagiert und auf verschiedene Weise ausgeführt werden kann, ist es wichtig, die Einstellungen zentral vornehmen zu können. Dazu wurde eine Konfigurationsdatei `config.json` erstellt, die alle verfügbaren Einstellungen auflistet. Diese wird zu Programmstart eingelesen und die Daten in ein Objekt der `Configuration`-Klasse geschrieben. Die einzelnen Komponenten können dann auf dieses Objekt zugreifen und die entsprechenden Werte verwenden. Zu den verfügbaren Einstellungen gehören unter Anderem:

[4] `https://docs.python.org/3/library/smtplib.html`

- Authentifizierung für die Twitter-API
- Authentifizierung für den SMTP-Server
- Empfänger der E-Mail
- Eine Liste von Suchtermen sowie die Suchsprache
- Den Pfad der `.csv`-Datei

Eine genauere Erklärung der jeweiligen Einstellungen findet sich in der mitgelieferten README-Datei.

3 Schlussfolgerung

Mit dem entwickelten Tool ist es möglich aktuelle relevante Tweets zu einem Suchbegriff zu finden. Die Qualität der Suchergebnisse ist in erster Linie durch die zur Verfügung stehenden Endpoints beschränkt. Durch unsere Priorisierung sind die Tweets, die einen hohen Prioritätswert erhalten haben, meistens tatsächlich relevant. Allerdings findet man auch weitere relevante Tweets, die von uns wenig priorisiert werden.

Um eine bessere Bewertung der Ergebnisse zu ermöglichen, wäre es möglich, eine Liste von Tweets als relevant oder nicht relevant zu markieren, und diese durch unseren Algorithmus priorisieren zu lassen. Anschließend könnte durch ein formales Kriterium entschieden werden, wie viel Fehlklassifikationen unser Programm erzeugt. Mit dieser Methode könnten auch die Gewichte, die zur Berechnung der Priorität verwendet werden, besser eingestellt werden. Dennoch kann eine rein statistische Herangehensweise nicht entscheiden, ob ein Tweet für das Thema Phishing relevant ist und ob in dem entsprechenden Kontext eine Antwort gerechtfertigt ist.

Für unsere Priorisierung wurde der eigentliche Text des Tweets nicht berücksichtigt. Mit einem anderen Algorithmus, der die Semantik des Textes beachtet, könnten möglicherweise relevante Tweets besser qualifiziert werden, wobei dieser Ansatz sicherlich weitaus komplexer ist.

Das Tool könnte auch durch weitere Funktionen ergänzt werden. Eine offensichtliche Erweiterung ist die vollständige Umsetzung des Webinterfaces. Durch eine bessere Integration mit dem Tool und der Twitter API wäre es möglich direkt über das Interface Antworten zu verfassen, ohne auf die Twitter-Website zu wechseln. Alternativ wäre es auch möglich, eine eigenständige GUI zu entwickeln.

Momentan basiert die Datenhaltung auf dem Importieren und Exportieren einer `.csv`-Datei. Dies ist im Code etwas umständlich, und führt auch beim Aktualisieren der Tweets zu einem vollständigen überschreiben der Datei. Es wäre von Vorteil, die Datenhaltung auf eine kleine Datenbank, beispielsweise SQLite zu übertragen. Der Dateiexport wäre natürlich weiterhin möglich.

References

1. Jansson, K., von Solms, R.: Phishing for phishing awareness. Behaviour & Information Technology **32**(6), 584–593 (2013). https://doi.org/10.1080/0144929X.2011.632650