2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) | 978-1-7281-1056-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/ASONAM49781.2020.9381432

Semantic Text Analysis for Detection of Compromised Accounts on Social Networks

Dominic Seyler, Lunan Li, and ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
{dseyler2, lunanli3, czhai} @illinois.edu

***Abstract*—Compromised accounts on social networks are regular user accounts that have been taken over by an entity with malicious intent. Since the adversary exploits the already established trust of a compromised account, it is crucial to detect these accounts to limit the damage they can cause. We propose a novel general framework for semantic analysis of text messages coming out from an account to detect compromised accounts. Our framework is built on the observation that normal users will use language that is measurably different from the language that an adversary would use when the account is compromised. We propose to use the difference of language models of users and adversaries to define novel interpretable semantic features for measuring semantic incoherence in a message stream. We study the effectiveness of the proposed semantic features using a Twitter data set. Evaluation results show that the proposed framework is effective for discovering compromised accounts on social networks and a KL-divergence-based language model feature works best.**

***Index Terms*—incoherence detection, semantic analysis, compromised accounts**

I. Introduction

The advent of social media has borne great opportunities and benefits, but also dangers and risk. One problem that has become more prominent is the spread of misinformation on social networks. In order to spread misinformation successfully, perpetrators mainly rely on social/spam bots [1, 2, 3] or compromised accounts [4, 5]. Account compromising has become a major issue for public entities and regular users alike. In 2013, over a quarter million accounts on Twitter were compromised [6] and despite massive efforts to contain account hacking, it is still an issue today [7]. For affected users a compromised account can be an embarrassing experience. As a result, 21% of users that fall victim to an account hack abandon the social media platform [4].

Compromised accounts are legitimate accounts that a malicious entity takes control over, with the intention of gaining financial profit [4] or spreading misinformation [8]. These accounts are especially interesting for attackers, as they can exploit the trust network that the user has established [9]. Since an account takeover can take up to five days, with 60% of the takeovers lasting an entire day [4], attackers are given ample time to reach their goal. Finding these hijacked accounts is challenging, since they exhibit traits similar to regular accounts [9]. Only after analyzing the changes in the account's behavior, patterns can be identified that expose the account as being compromised and therefore specific methodology is required.

IEEE/ACM ASONAM 2020, December 7-10, 2020
978-1-7281-1056-1/20/$31.00 © 2020 IEEE

Existing work on detection of compromised accounts has mostly relied on anomalies of user profiles, but there is a great opportunity to leverage semantic analysis of an account's content, since the intent of compromising a social media account is to inject "abnormal" content [10, 11, 12]. Thus, analysis of semantic coherence of text content can be a general strategy for detecting compromised accounts. While some existing work has attempted to leverage text content also, the textual features used are simplistic or non-interpretable [8, 10, 13]. For example, bag-of-word features are inadequate for capturing semantic variations in language, while embedding-based approaches are not interpretable, which is important if such a method is to be used to guide any real-world actions on the account. In this paper, we propose to perform deeper semantic analysis using a solid probabilistic language model framework to directly measure the semantic incoherence in text content, leading to highly interpretable semantic features for detecting compromised accounts. Such features can be used in any supervised learning framework to improve detection accuracy and improve explainability.

Our key observation is that a regular user's textual output will differ from an attacker's textual output. We thus propose a general framework for detecting compromised accounts based on semantic analysis of the incoherence in the text stream. This is complementary with the existing work in the sense that our framework can lead to highly interpretable novel features that can be added to any existing machine learning-based detection method to improve its accuracy and explainability. As a specific implementation of the framework, we model the user's and attacker's language as two smoothed multinomial probability distributions that are estimated using the textual output of the user and attacker, respectively. We use a similarity measure between probability distributions as indicators that an account is being compromised. Even though we do not know the start and the end of an account takeover, our method leverages the fact that the average difference for random account begin and end dates will be higher for compromised accounts, compared to benign accounts.

Evaluation of such a detection task is challenging due to the inevitable concern of user privacy, making it nearly impossible to have a publicly available real world data set. Following other work in this domain [9, 12, 14, 15], we propose a simulation-

based evaluation method and use such a method to study the effectiveness of the proposed semantic analysis method. Specifically, we formalize this problem in a threat model and utilize this model to simulate account takeovers in our dataset, to compare an implementation of the proposed framework[1] to other approaches.

Our evaluation results show that the proposed incoherence-based features are highly effective for detecting compromised accounts and can be combined with other features to improve detection accuracy and enhance explainability. We further show that when the proposed method is trained on simulated data, it can detect non-artificially compromised accounts from real world data set. Since training on simulated data requires no human effort, the proposed method can be immediately adopted by a social media company to potentially enhance their compromised account detector.

II. Related Work

It is common practice in compromised account detection to build profiles based on user behavior and look for anomalies within them. Egele et al. [8] learns behavioral profiles of users and looks for statistical anomalies in features based on temporal, source, text, topic, user, and URL information. Ruan et al. [9] finds anomalies in the variance of a user's click behavior. Viswanath et al. [16] applies principal component analysis to a user's Facebook likes to find abnormal behavior. Vandam et al. [13] studies certain account characteristics, such as number of hashtags or number of mentions in tweets, which are used as features in a classification framework. Karimi et al. [10] uses Long Short-Term Memory networks to capture the temporal dependencies within user accounts to learn distinguishing temporal abnormalities. VanDam et al. [11] uses an unsupervised learning framework, where multiple views on a user profile (i.e., term, source, time and place) are encoded separately and then mapped into a joint space. This joint representation is then used to retrieve a ranking of compromised accounts. Building on this work, VanDam et al. [12] uses and encoder-decoder framework to build low-dimensional feature vectors for users and tweets. The residual errors form both encoders are used in a supervised setting to predict compromised accounts.

When it comes to textual information, current methods either do not leverage it at all [8, 9] our the textual features are superficial. For instance, text is only used to detect language changes (e.g., from English to French) [8] or topics are derived plainly from hashtags [8, 13]. It is also common to simply use bag-of-word features [13] or neural embeddings [10, 11, 12]. Thus, all methods can benefit of a deeper analysis of the semantic incoherence of text.

III. Semantic Incoherence Framework

A. Threat Model

The adversary's goal is to inject textual output into a benign account in order to mask the output's origin and leverage the user's influence network. In our setting, it is irrelevant how the adversary obtained access to the account. To make our method as general as possible, we assume that no account details are observed, meaning that we ignore the friendship network, profile details, etc. To further cater to generality, we don't make any assumptions about the text, except that it was written by a different author. In the case of an attack, we assume that at least one message is injected by the adversary and that at least one benign message is observed. For all accounts we assume that they existed for more than one day.

[1]The code is available at: https://github.com/dom-s/comp-account-detect.

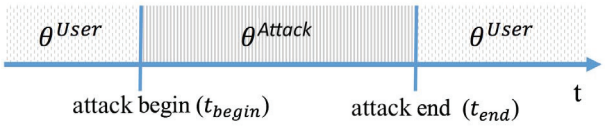


Fig. 1: A tweet stream divided according to begin (t_{begin}) and end (t_{end}) of an attack. A language model is learned for regular tweets (θ^{User}) and compromised tweets (θ^{Attack}).

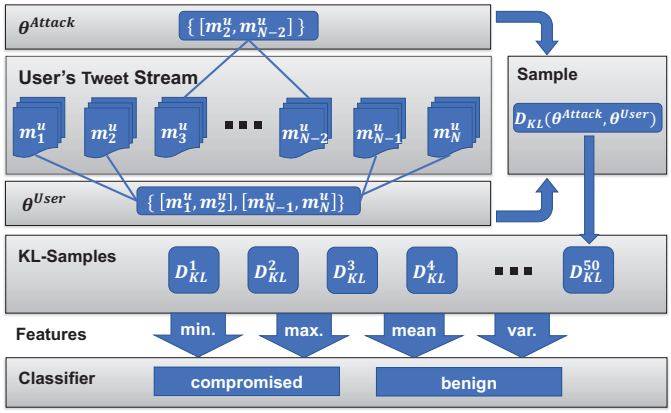


Fig. 2: Overview of model.

B. General Framework

We now describe the proposed framework for identifying compromised accounts. In accordance with our threat model, the framework is based on the assumption that an adversary's textual output will deviate from a regular user's textual output. Let $\mathcal{U}$ be the set of all users u. Further, let m_t^u be a message m from user u at time $t \in \{1, 2, ..., N\}$. Our goal is to find all compromised user accounts $\mathcal{U}^{comp} \subset \mathcal{U}$. To capture the discrepancy between language usage, we propose to divide the tweet space of a user into two non-overlapping sets M^{User} and M^{Attack}. We randomly assign two timepoints: t_{start} signals the start of the attack and t_{end} signals its end. All m_t^u with $t \in [t_{start}, t_{end}]$ make up M^{Attack}, whereas all m_t^u with $t \in [t_1, t_{start-1}] \cup t \in [t_{end+1}, t_N]$ make up M^{User}. We can measure the difference between M^{User} and M^{Attack} using any similarity measure of our choice. This procedure can be repeated multiple times for different values of t_{start} and t_{end}. t can be of different granularity, where the minimum is per-

message and the maximum can be chosen at will. The more often the procedure is repeated for a certain user, the higher is the sampling rate, which makes for better approximations of the true difference between M^{User} and M^{Attack} (we study the optimal sample rate in Section V). Thus, this strategy provides a flexible trade-off between accuracy and efficiency. The similarity measures can then be employed as features in the downstream task.

C. Instantiation with Language Modeling

We describe our practical instantiation of the framework using language modeling and supervised learning. We create a classifier that distinguishes compromised from benign user accounts based on a feature set, derived from our framework. We measure this as the difference between two word probability distributions of a user and an attacker. We select KL-divergence [17] as our method of choice to compare the difference of probability distributions. We assume that when a user writes a message she draws words from a probabilistic distribution that is significantly different from the distribution an attacker draws words from. Let θ^{User} and θ^{Attack} be two word probability distributions (i.e., language models) for the user and attacker. We need to select two time points t_{start} and t_{end} that mark beginning and end of the attack (Figure 1). As described by our framework, all messages m_t^u that fall within the time interval $[t_{start}, t_{end}]$ contribute to θ^{Attack}, whereas $m_t^u \notin [t_{start}, t_{end}]$ contribute to θ^{User}.

Figure 2 presents an overview of our model. For a particular user, we sample her tweet stream for random t_{start}, t_{end} pairs. As seen in the figure, the algorithm can, for example, select t_3 and t_{N-2} as t_{start} and t_{end}, respectively. Thus, all tweets that fall between $\{[m_1^u, m_2^u], [m_{N-1}^u, m_N^u]\}$ contribute to θ^{User}. All tweets that fall in $[m_3^u, m_{N-2}^u]$ contribute to θ^{Attack}. Then, the KL-divergence for these specific θ^{User} and θ^{Attack} contributes one sample for this user (D_{KL}^{50} in the figure). This process is repeated for different samples (e.g., we used 50 samples in our experiments), where each time t_{start} and t_{end} are selected at random, with constraint $t_{start} < t_{end}$. Sample rates are selected empirically, depending on the best classification performance on a development set. We select the maximum, minimum, mean and variance of the sampled KL-divergence scores. These features are then combined in a Support Vector Machine (SVM) model, which learns the optimal weighting for each feature based on the training data. Naturally, other classifiers can also be used, but exploration of different classifiers is out of the scope of this paper.

D. Language Modeling Details

We try to estimate the joint probability of $P(w_1, w_2, w_3, ..., w_N)$ for all words $w_1, ..., w_N$ in the text. According to the chain-rule, this is equivalent to computing $P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_N|w_1, ..., w_{N-1})$.
Because of the combinational explosion of word sequences and the extensive amount of data needed to estimate such a model, it is common to use n-gram language models. N-gram language models are based on the Markov assumption that a word w_1 only depends on n previous words ($P(w_k|w_1, ..., w_{k-1}) \approx P(w_k|w_{k-n+1}, ..., w_{k-1})$). The simplest and computationally least expensive case is the uni-gram. Here, the Markov assumption is that a word w_k is independent of any previous word, i.e., $P(w_k|w_1, ..., w_{k-1}) \approx P(w_k)$. All $P(w_k)$ for $k \in 1, 2, ..., N$ make up a language model θ, which is a multinomial probability distribution, where words in the document are events in the probability space. The parameters for the language models θ are estimated using maximum-likelihood. Maximizing the likelihood of the uni-gram language model is equivalent to counting the number of occurrences of w_k and dividing by the total word count ($P(w_k) = \frac{c(w_k)}{N}$, where $c(w_k)$ is the word count of w_k). This distance between two probability distributions can be estimated using Kullback-Leibler-divergence [17]. In the discrete case, for two multinomial probability distributions P and Q the KL-divergence is given as

$$D_{KL}(P, Q) = \sum_i P(i) log(\frac{P(i)}{Q(i)}).$$

It can be observed that $D_{KL}(P, Q) \neq D_{KL}(Q, P)$. However, it is common practice to still think of D_{KL} as a distance measure between two probability distributions [18]. An issue with D_{KL} is that the sum runs over i, which is the event space of P and Q. Thus, it requires the event space to be equivalent for both distributions. In our case θ is a language model. Now, let $v(\theta)$ denote the vocabulary set of θ. As a result of maximum-likelihood estimation, in most cases $v(\theta^{User}) \neq v(\theta^{Attack})$. Thus, we have to smooth the probability distributions such that $v(\theta^{User}) = v(\theta^{Attack})$, which is required in order to calculate D_{KL}. To achieve this, we define $v(\theta) := v(\theta^{User}) \cup v(\theta^{Attack})$. Then, we set $v(\theta^{User}) = v(\theta)$ and $v(\theta^{Attack}) = v(\theta)$ and estimate θ^{User} and θ^{Attack} using the Laplace estimate.

IV. Experimental Design

We turn the compromised account detection task into a binary classification problem, where the goal is to decide whether a user account is compromised or not. We therefore learn a classification function, which returns 1 if an account is compromised and 0 otherwise.

We set up our experiments to answer the following research questions: We first perform a feasibility analysis to **(I)** study to what extend a KL-divergence measure can detect incoherence. The second part of our feasibility analysis finds proof that **(II)** the average KL-divergence can be estimated by randomly sampling a certain number of points with different begin/end dates. Following the feasibility study, we show how effective the proposed language model based method is in detecting compromised accounts in a simulated environment. We show **(III)** how the proposed features compare to general state-of-the-art text classification features and how to combine them; **(IV)** how our proposed features perform in comparison to other compromised account detection methods and how we can further improve performance by combining their feature

spaces; **(V)** how effective is our method on a real (non-simulated) dataset by performing a qualitative, manual investigation. We start by introducing our dataset.

A. Dataset

Previous methods on compromised account detection either do not publish their datasets [8, 13, 16] or the original text data is not fully recoverable due to restrictions of the social media platform or the deletion of user data [10]. Therefore, meaningful evaluation is especially challenging in our setting. Since no dataset exists, we opted to follow existing practise [9, 12, 14, 15] and use a simulation method. We perform a simulation of account attacks, which allows us to compare different methods quantitatively to study their effectiveness. Any bias introduced by the simulation is unlikely to affect our conclusions as it is orthogonal to the methods that we study and we argue that such an evaluation strategy is adequate to perform a fair comparison of different methods.

For simulation, we leverage a large Twitter corpus from Yang and Leskovec [19]. The dataset contains continuous tweet streams of users. Relationships between users, etc. are unknown. Finding compromised accounts manually within a dataset of millions of tweets is clearly infeasible. Simulating account hijackings enables us to (cheaply) create a gold standard while having full control over the amount of accounts compromised and begin/end of account takeovers. For simulation, we follow our threat model introduced in Section III-A, where the adversary's goal is to inject messages into regular accounts. Since we make no assumptions about the textual content, we follow a similar methodology to VanDam et al. [12] and Trang et al. [15], and replace part of the consecutive tweet stream of an account with the same amount of consecutive tweets from another random user account. Since our algorithm has no knowledge about the begin and end of an attack, we choose these at random, meanwhile ensuring that only a certain predefined fraction of the tweets is compromised. This allows us to test the effectiveness of our method in different scenarios, where different percentages of tweets are compromised within an account. To get meaningful estimates for the language models θ^{User} and θ^{Attack}, we select the 100,000 users with most tweets in our data. According to our threat model, we retain users with more than one day of coverage, resulting in 99,912 users and 129,442,760 tweets.

B. Feasibility Studies

In our feasibility studies we use a subset of the data by selecting 495 users at random, where each compromised account contains 50% compromised tweets. For each user, we put all tweets into daily buckets and calculate the KL-divergence for all possible combinations of t_{begin} and t_{end}. The reason this cannot be done for the whole dataset is simply because of computational cost. We therefore operate on this subset to investigate whether our method is feasible and can be approximated to avoid increased computational effort.

C. Quantitative Evaluation

Our quantitative evaluation aims to show the performance of the algorithm using standard metrics for performance measure. We measure classification Accuracy; Precision, Recall and F_1-score for the class representing the predictions of compromised accounts (1-labels). In addition, we show effectiveness of our method for different levels of difficulty. We experiment with various settings for the percentage of compromised tweets in compromised accounts with random begin and end dates of account take over. More concretely, we select 50%, 25%, 10% and 5% of tweets to be compromised, each representing a more difficult scenario. As these ratios would not be strictly separated in a real-world scenario, we further experiment with random ("RND") ratios, drawn uniformly from [5%,50%]. In our experiments the probability of an account being compromised is set to 0.5 to obtain a balanced dataset[2]. Furthermore, we employ ten-fold cross validation.

D. Baselines

To study the effectiveness of our proposed features based on language modeling (*LM*), we compare them to general text classification features (i.e., word-based and Doc2Vec [20]) and task-specific features from two methods in compromised account detection, namely COMPA [8] and VanDam [13]. We combine them in a Support Vector Machine (SVM) framework, which has been shown to be an excellent predictor for text classification. However, our approach is general and can be applied to any classifier.

SVM. We use SVM with linear kernels for all models. Features are standardized by removing the mean and scaling to unit variance. Ten-fold cross validation is performed for all models to ensure there is minimal bias in selecting the training/testing split of the data. All posts of a single user are concatenated as one document. The labels for each document were chosen as 1 if an account is compromised and 0 otherwise.

Word-based Features. For word-based feature representation and feature selection we follow the methodology of Wang et al. [21]. We choose count based (*COUNT*) and TF*IDF based (*TF*IDF*) representations of words and experiment with two supervised feature selection strategies, namely chi-square and mutual information. For dictionary creation we keep 100,000 uni-grams, which appear in no less than 20 and no more than 9991 (=10%) of documents, to remove very rare and very common words.

Document-based features (*Doc2Vec*). We leverage the method proposed in Le and Mikolov [20] to learn low-dimensional vector representations for documents. As recommended, we choose a vocabulary size of 2M tokens [22], 400-dimensional document vectors and a window size of 5 [20]. We make use of the document vectors as features in our SVM classification framework.

[2]In reality the amount of compromised accounts is much lower then 50%. However, it is common in supervised frameworks for compromised account detection to balance datasets to learn a better discriminative function [10, 12].

COMPA [8]. This method builds behavioral user profiles and detects compromised accounts by finding statistical anomalies in the features. We implement features that can be derived from our dataset: time (hour of day), message text (language), message topic, links in messages and direct user interaction. This method is used for stream processing; once a user profile is build, the method checks a new message against the existing user profile. For each feature the message will be assigned an anomaly score, which reflects how different this message is from the previously observed user profile. Since our method classifies accounts as a whole, we simulate this process for each message within an account (in temporal order of the posting of the message). We then aggregate the anomaly scores, per feature, as the mean of the scores of all the account's messages.

VanDam [13]. This work investigated the distributional properties of different features on compromised accounts and uses them in a classification framework. We implement the following features from this method: hashtags, mentions, URLs, retweets and sentiment. The authors use these features to classify single messages as compromised. In our setting, we aggregate each feature's counts over all messages within an account and use the mean of the counts as features.

E. Qualitative Evaluation

Our qualitative evaluation aims to show how well compromised accounts can be detected in the original dataset. Therefore, we manually evaluate the accounts that have the highest probability according to our classifier. Here, a major change is that we only inject messages into the *training* dataset, whereas the testing set is left untouched. We randomly select 70% of users for training and 30% of users for testing, with 25% of tweets compromised. Since there is no ground truth in the testing dataset, we define four specific metrics to evaluate whether an account is considered compromised. If two or more of the following metrics are met, an account is considered as being compromised: In certain time periods the account (1) has a sharp topic change, (2) has a specific posting frequency change, (3) has a specific language change (4) posts repeated tweets.

V. Experiment Results

A. Language Model Similarity of Compromised Accounts

In the first part of our feasibility analysis we answer research question **(I)** and study to what extend a KL-divergence measure can detect incoherence.

To achieve this, we manually inspect the differences in user accounts by leveraging a heatmap as a visual cue. Figure 3 depicts four heatmaps of two benign (Figures 3a, 3b) and compromised user accounts (Figures 3c, 3d). The x-axis of each figure depicts different values for t_{begin}, while the y-axis depicts different values for t_{end}. The color palette ranges from blue (low KL-divergence) to red (high KL-divergence). The left part of the plot below the diagonal is intentionally left empty, since these values would represent nonsensical variable assignments for t_{begin} and t_{end}.

It is immediately evident that we find more high KL-divergence values for compromised compared to benign accounts. In the figures, high values are expressed by red and dark red colors. From this observation it can be inferred that the average KL-divergence will be higher for accounts that are compromised. By manually inspecting over 100 of these user account heatmaps we find that many of them follow this general trend. We understand this as preliminary evidence that a method which utilizes KL-divergence for detecting compromised accounts is feasible.

We further noticed in Figures 3c and 3d that the account takeover happened where the KL-divergence reaches its maximum (see the dark circle in Figure 3c and tip of the pyramid in Figure 3d). Unfortunately, this is not true for all inspected accounts, but it gives reason to believe that there might be potential to find the most likely period of an account takeover with our current framework. This could be done by finding t_{begin} and t_{end} that maximizes the difference of the language models θ^{User} and θ^{Attack}.

B. Estimate KL-divergence Using Random Sampling

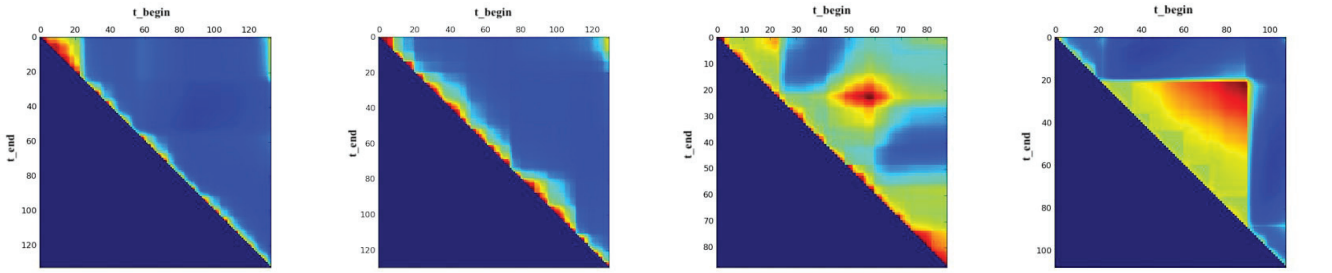
In our second feasibility analysis we investigate research question **(II)**, whether the average KL-divergence of a user account can be approximated using random sampling. More specifically, we try to find a reasonable estimate by calculating the KL-divergences only for a subset of the t_{begin} and t_{end} pairs.

Since we have calculated the KL-divergence for every possible combination of t_{begin} and t_{end} for all of the 495 users, we know the actual average KL-divergence. We then try different sampling rates. For every sampling rate we average over the samples and compare them to the actual average that is calculated using all t_{begin}, t_{end} pairs. The result is shown in Figure 4. In the figure we plot the actual average KL-divergence for compromised and benign against the averaged samples for different sample rates. The sample rates range from 1 to 121. The plot shows that the average KL-divergence for compromised accounts is about 0.1 higher than for benign accounts. This confirms our findings in Section V-A. Furthermore we see that for small sample rates (< 81) there are minimal deviations for the average (± 0.01). The higher the sample rate the lower these deviations become, as our estimate gets better.

Since our estimates only deviate slightly we also investigate the mean squared error (*mse*). Here, the mse is defined as: $\frac{1}{n}\sum_{u \in U^{test}}(sampled\_avg(u) - actual\_avg(u))^2$. In Figure 5 we plot the mse for compromised and benign accounts. For very small sampling rates (< 50) we see errors of over 0.07 and over 0.06 for compromised and benign accounts, respectively. Once the sample rate is greater than 101 the mse is close to 0. We therefore conclude that a sampling rate in the range of [50, 100] is sufficient for our experiments.

C. Effectiveness on Simulated Data

In this subsection we show the effectiveness of our language model based method when detecting compromised accounts



(a) Benign user account (b) Benign user account (c) Compromised user account (d) Compromised user account

Fig. 3: KL-divergence heatmap for different benign (Figure 3a and 3b) and compromised user accounts (Figure 3c and 3d). The x-axis of each figure depicts different values for t_{begin}, while the y-axis depicts different values for t_{end}. The color palette ranges from blue (low KL-divergence) to red (high KL-divergence).

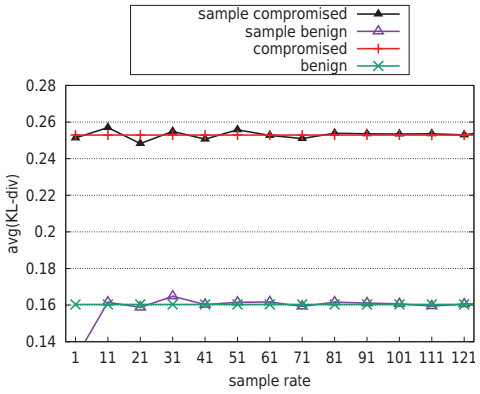


Fig. 4: Average KL-divergence for different sample rates.

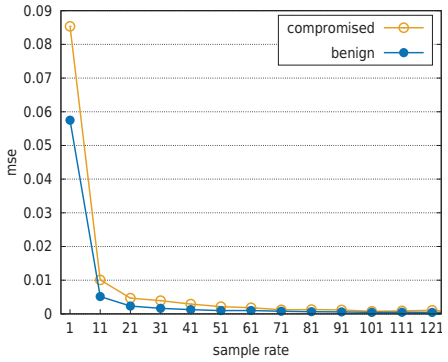


Fig. 5: Mean squared error (mse) for different sample rates.

in a simulated environment. We answer research question **(III)**, compare the performance to general state-of-the-art text classification features and **(IV)**, show that improvements can be made when combining the language model features with other specialized approaches, proposed in previous work in compromised account detection.

We perform an ablation study in Table I, which shows the performance of all features together and individually. In this experiment 50% of the tweets of a compromised account are compromised. We observe that we gain maximum performance over all metrics when all features are utilized (see column

TABLE I: Ablation study using different measures.

Measure	All	Max	Min	Mean	Var.
Accuracy	**0.80**	0.59	0.76	0.75	0.48
F_1	**0.78**	0.57	0.72	0.72	0.35
Precision	**0.90**	0.61	0.87	0.83	0.47
Recall	**0.68**	0.53	0.61	0.64	0.27

"All"). The classifier reaches an accuracy of 0.80, high precision (0.90) and Recall at 0.68. Both measures are combined in the F_1-score, which reaches 0.78. We would like to note that a precision of 0.90 can be sufficient for many practical applications, where the system could alert users or platform providers. When features are investigated individually, we find that the features "Max" and "Variance" perform worst. The features "Min" and "Mean" perform almost equally in isolation and much better than the other two. We find the minimum sample of the two probability distributions to be a strong signal. This further confirms our earlier observations that compromised accounts can be distinguished by their higher average KL-divergence.

We turn to research question **(III)** and investigate how our language model based features compares to state-of-the-art text classification features. Table II lists the results for our features (*LM*) and *Doc2Vec*. Best performance is reached when 50% of tweets are compromised, with accuracy at 0.80 and 0.71, respectively. When the number of compromised tweets is reduced by half, the accuracy drops to 0.75 and 0.69. When the amount of compromised tweets is set to 10% and 5%, accuracy further reduces for both features. For random ratios, the performance is slightly lower than for the fixed 25% ratio. It is expected that this performance falls somewhere within the lower (5%) and upper (50%) bounds for performance. Summarizing, we see that *LM* achieves higher Accuracy and Precision, whereas *Doc2Vec* achieves higher Recall.

In Table III we compare *LM* to general text classification features and their combinations. It can be seen that *Doc2Vec* performs better than the word based models. This might be due to the fact that *Doc2Vec* is able to learn a better representation for each document, compared to *TF*IDF* and *COUNT*. *Doc2Vec* outperforms the *TF*IDF* model by up to 15 percentage points (≈27% relative improvement), if the

TABLE II: *LM* and *Doc2Vec* features, with different percentages of compromised tweets.

	LM				*Doc2Vec*			
%	Accuracy	F_1	Precision	Recall	Accuracy	F_1	Precision	Recall
50	**0.80**	**0.78**	**0.90**	0.68	0.71	0.71	0.72	**0.71**
25	**0.75**	**0.71**	**0.82**	0.63	0.69	0.69	0.70	**0.69**
10	**0.65**	0.62	**0.69**	0.56	0.63	**0.63**	0.64	**0.63**
5	**0.59**	0.56	**0.60**	0.52	**0.59**	**0.59**	0.59	**0.59**
RND	**0.75**	**0.70**	**0.81**	0.61	0.68	0.68	0.69	**0.68**

TABLE III: Accuracy for different features and their combinations. Our method (*LM*) is compared to general text representations.

%	*COUNT*	*TF*IDF*	*Doc2Vec*	*Doc2Vec + TF*IDF*	*LM*	*LM + TF*IDF*	*LM + Doc2Vec*	all
50	0.53	0.56	0.71	0.72	0.80	0.81	**0.87**	**0.87**
25	0.53	0.55	0.69	0.69	0.75	0.75	**0.82**	**0.82**
10	0.52	0.54	0.63	0.63	0.65	0.65	0.70	**0.71**
5	0.52	0.53	0.59	0.59	0.59	0.59	**0.62**	**0.62**
RND	0.53	0.55	0.68	0.68	0.74	0.74	**0.80**	**0.80**

amount of compromised tweets reaches 50%. The performance improvement is less drastic when only small amounts of tweets are compromised. We further find that *LM* performs best, as it outperforms *TF*IDF* and *Doc2Vec* with 6 and 7 percentage points when only 5 percent of tweets are compromised, respectively. The most distinguishing performance is achieved when the amount of compromised tweets reaches 50%. There, *LM* outperforms *Doc2Vec* with 9 percentage points (≈13% relative improvement) and *TF*IDF* with 24 percentage points (≈43% relative improvement). We argue that this is strong evidence for the superiority of the *LM* feature compared to other general text classification features.

When combining features, we find that adding *Doc2Vec* features to *LM* results in the highest performance improvements, with up to 7 percentage points over *LM* alone (column: "*LM + Doc2Vec*"). Adding *TF*IDF* to *LM* or *Doc2Vec* does only have a negligible effect (columns: "*LM + TF*IDF*" and "*Doc2Vec + TF*IDF*"). The same minimal improvements are observed when adding *TF*IDF* to *LM* and *Doc2Vec* (column: "all"). Thus, we conclude that the proposed *LM* features add meaningful signals to general text classification features.

D. Comparison with Existing Detection Methods

For answering research question **(IV)**, we show how *LM* performs in comparison to other compromised account detection methods and that improvements can be made by combining *LM* with these methods. The baseline features here differ in the way that they are specifically designed for compromised account detection, whereas the previous features were general textual representations.

We compare all models in Table IV. The first three rows compare the compromised account detection features on their own. Our model outperforms the strongest baseline (COMPA) over all metrics, with the highest gains made in accuracy and precision, increasing 19.4% and 26.6%, respectively. The fifth through seventh row show the combination of the *LM* features with the two baselines alone and in combination. Here we find that our method can further improve when the baseline features are added. We argue that our features are orthogonal to existing work, as evidenced by the performance improvement due to combination of the feature spaces. The second-to-last row shows the performance, when all baseline features are combined with the *LM* model. This results in the best performing model, with gains over all metrics.

E. Effectiveness on Real Data

The final question we answer qualitatively is **(V)**, how effective is our method on non-simulated data. We sort 20 accounts with the highest probability of being compromised into six categories, shown on the left in Table V. We find that most of these accounts belong to categories with high variation in language, i.e., news, spam, re-tweet bot. One of the accounts is found to be compromised. This result needs to be seen in perspective, since the classifier was trained on simulated data. We would expect the percentage of compromised accounts to be much higher, if training data with "real" labels are used. These results show that our algorithm can also detect "unusual" accounts and users, thus potentially enabling development of novel text mining algorithms for analyzing user behavior on social media, which should be an interesting future research topic.

We further inspect the current state of each account on the right side of Table V. We find that most accounts are abandoned and one account was suspended by Twitter. The account that was identified as compromised was set to protected, which could be an indicator that this user had become more conscious about tweets that were posted from her account and therefore decided to not share her tweets publicly. While manually investigating the account's tweets, we find that after discussing general topics a near-duplicate message is posted hundreds of times with only brief pauses between tweets. Different users were addressed directly, which were most likely followers of the account. With this scheme the attacker tries to directly grab the attention of a targeted user. The messages included one of two links that were identified as suspicious by the link-shortening service the hacker utilized to hide the actual URL. After the attack, the tweets return to discuss similar topics as before. From the content of this messages we conclude that the hacker was pursuing a led generation scheme [4], where users are lured into clicking a link. It is reasonable to assume

TABLE IV: Comparison to features from related methods. The dataset has random percentages of tweets compromised (RND).

Model	Accuracy	F_1	Precision	Recall
COMPA [8]	0.62	0.60	0.64	0.56
VanDam [13]	0.50	0.47	0.50	0.45
LM	**0.74**	**0.70**	**0.81**	**0.61**
improvement *LM* over best baseline	19.4%	16.7%	26.6%	8.9%
LM + COMPA	0.75	**0.73**	0.81	0.66
LM + VanDam	0.74	0.71	**0.82**	0.62
LM + COMPA + VanDam	**0.76**	**0.73**	0.81	**0.67**
improvement over *LM*	2.7%	4.3%	1.2%	9.8%
LM + *Doc2Vec* + *TF*IDF* + COMPA + VanDam	**0.81**	**0.79**	**0.85**	**0.75**
improvement when adding standard features	6.6%	8.2%	4.9%	11.9%

TABLE V: Statistics of manually evaluated accounts.

Category	Count	Status	Count
News	5	Abandoned	7
Spam	4	Active	6
Re-tweet Bot	2	Deleted	4
Compromised	1	Protected	2
Regular	7	Suspended	1
Unknown	1		

that if our algorithm were applied at much larger scale to all the Twitter users, it would most likely be able to detect many more compromised accounts.

VI. Conclusion and Future Work

We proposed a novel general framework based on semantic text analysis for detecting compromised social media accounts. Following the framework, we proposed a specific instantiation based on uni-gram language models and KL-divergence measure, and designed features accordingly for use in a classifier that can distinguish compromised from benign accounts. We conclude that (1) the proposed *LM* feature is most effective, even when used as a single feature-based detection method. (2) *LM* captures new signals that haven't been captured in the existing methods and features, which is shown by the further improvement when added on top of the baselines. (3) The best performing method would combine the proposed LM with all the existing features. Although *LM* is motivated by a security problem, our general idea of performing differential semantic analysis of text data may be applicable to other domains where incohesion (or outlier) in text data needs to be captured.

References

[1] S. Cresci *et al.*, "Better safe than sorry: an adversarial approach to improve social bot detection," in *Web Sci'*, 2019.

[2] C. Grier, K. Thomas, V. Paxson, and C. M. Zhang, "@spam: the underground on 140 characters or less," in *CCS*, 2010.

[3] J. Martinez-Romo and L. Araujo, "Detecting malicious tweets in trending topics using a statistical analysis of language," *Expert Syst. Appl.*, 2013.

[4] K. Thomas *et al.*, "Consequences of connectivity: Characterizing account hijacking on twitter," in *SIGSAC*, 2014.

[5] E. Zangerle and G. Specht, ""sorry, I was hacked": a classification of compromised twitter accounts," in *SAC*, 2014.

[6] H. Kelly, "Twitter hacked; 250,000 accounts affected," February 2013. [Online]. Available: https://cnn.it/2XIpLYZ

[7] K. Olmstead and A. Smith, "Americans and cybersecurity," *Pew Research Center*, vol. 26, 2017.

[8] M. Egele *et al.*, "COMPA: detecting compromised accounts on social networks," in *NDSS*, 2013.

[9] X. Ruan *et al.*, "Profiling Online Social Behaviors for Compromised Account Detection," *IEEE Trans. Information Forensics and Security*, 2016.

[10] H. Karimi *et al.*, "End-to-end compromised account detection," in *ASONAM*, 2018.

[11] C. VanDam *et al.*, "Cadet: A multi-view learning framework for compromised account detection on twitter," in *ASONAM*, 2018.

[12] ——, "You have been caute! early detection of compromised accounts on social media," in *ASONAM*, 2019.

[13] ——, "Understanding compromised accounts on twitter," in *WI*, 2017.

[14] S. Gupta *et al.*, "Modeling and detecting anomalous topic access," in *ISI*, 2013.

[15] D. Trang *et al.*, "Evaluating Algorithms for Detection of Compromised Social Media User Accounts," in *ENIC*, 2015.

[16] B. Viswanath *et al.*, "Towards detecting anomalous user behavior in online social networks," in *USENIX*, 2014.

[17] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Ann. Math. Statist.*, 1951.

[18] C. Zhai and S. Massung, *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Morgan & Claypool, 2016.

[19] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *WSDM*, 2011.

[20] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014.

[21] Y. Wang *et al.*, "A study of feature construction for text-based forecasting of time series variables," in *CIKM*, 2017.

[22] J. Pennington *et al.*, "Glove: Global vectors for word representation," in *EMNLP*, 2014.