Interpretability of API Call Topic Models: An Exploratory Study

Puntitra Glendowne *
Stephen F. Austin State University
sawadponp@sfasu.edu

Dae Glendowne *
Stephen F. Austin State University
glendowndj@sfasu.edu

Abstract

Topic modeling is an unsupervised method for discovering semantically coherent combinations of words, called topics, in unstructured text. However, the human interpretability of topics discovered from non-natural language corpora, specifically Windows API call logs, is unknown. Our objective is to explore the coherence of topics and their ability to represent the themes of API calls from the perspective of malware analysts. Three Latent Dirichlet Allocation (LDA) models were fit to a collection of dynamic API call logs. The topics were manually evaluated by malware analysts. The results were then compared to existing automated quality measures. Participants were able to accurately determine API calls that did not belong to topics learned by the 20 topic model. Our results agree with topic coherence measures in terms of highest interpretable topics. The results are not compatible with log-perplexity, which concur with the findings of topic evaluation literature on natural language corpora.

1. Introduction

The total number of new malware has increased significantly in the past several years from less than 100 million in 2012 to over 900 million in 2019 [1]. Automated processes to analyze large collections of malware are vital as manual investigation is time consuming to cover this rapidly growing quantity of malware. While the use of clustering algorithms to group malware based on dynamic and/or static features has been studied, simply clustering malware using features extracted from static and dynamic analyses does not provide insights into how malware are actually grouped together. Features are usually difficult to understand or the number of features used are too large to be understood by humans.

As malware analysts continue to struggle with an ever growing quantity of malware to analyze, they would benefit from automated systems that expedite the malware characterization process by providing meaningful summaries of malware behavioral themes that assist further in-depth analyses. The development of a model that captures the key behavioral themes could help alleviate some of the time burden from human malware analysts.

Probabilistic topic models have been widely explored in the literature and used in many applications [2, 3, 4]. Topic modeling is a group of unsupervised, generative methods for characterizing unstructured text with semantic topics [5]. Topic models have been studied mainly with general natural language corpora (i.e. Wikipedia, movie archives, and newspaper articles). Existing studies have evaluated the interpretability of topics with collections of documents in natural language [3, 6, 7] but the interpretability of topics obtained from application programming interface (API) call logs has not been quantitatively evaluated with human subjects. In the cybersecurity domain, similarities between natural language and API calls are recognized [8, 4, 9]. Each API call log consists of a series of API calls the same way a document consists of series of words. Each word has meanings depending on the context around it or how it is used. For example, NtClose, which is a routine used for closing object handles, can be used with different object such as files or registry keys. Additionally, some API calls could be synonyms of other API calls (i.e. NtWriteFile and ZwWriteFile) the same way words in natural language could be synonyms. In our context, a topic can be perceived as a behavioral theme, where each topic represents semantically coherent combinations of API calls (words). For example, DLL injection behavior can be represented by the combination of NtOpenProcess, NtAllocateVirtualMemory, NtWriteVirtualMemory, and CreateRemoteThread API calls.

With similar characteristics between natural language corpora and API call log collection, topic modeling has been applied to generate topics as features

* Work done while at Distributed Analytics and Security Institute

URI: https://hdl.handle.net/10125/64535
978-0-9981331-3-3

for the purpose of classification [4, 9]. Providing behavioral themes of malware directly to malware analysts is a possible mechanism for automatic characterization and information filtering of malware behaviors. However, doing so would require topics to be human interpretable and accurately reflect the actual behaviors of malware. Currently, no studies have investigated the interpretability of topic models trained from API call logs. Our study addresses this research gap by exploring the interpretability of topics trained with Windows API call logs and evaluated by human subjects. To understand Windows API call logs, one has to have domain knowledge of Windows malware reverse engineering. Specifically, they need to understand what each API call does and which API calls tend to appear together for certain behaviors. Therefore, our study requires human assessors with specialized domain knowledge in Windows malware analysis.

In this paper, we explore the interpretability of topic models of Windows API calls with different numbers of topics by human subjects with a background in Windows malware analysis. We also compare the results with automated quality measures of topic model goodness and topic coherence. This study serves as a preliminary baseline for applying topic modeling with non-natural language corpora, specifically API call logs. Our main goal is to show that topic modeling is able to reveal semantically coherent behavioral themes that are representative of Windows API call logs.

The rest of this paper is organized as follows. Section 2 highlights background and related work. Section 3 explains the study methodology, tools used for collecting data, study subjects, and the quality measures computed and used in this study. Section 4 presents the quantitative results obtained as well as qualitative and quantitative analysis of the results. Limitations of the study are addressed in Section 5. Finally, section 6 summarizes the results, gives conclusions of the study, and outlines possible future work.

2. Background

2.1. Malware Analysis

Malware analysis techniques are often divided into two categories: static or dynamic. Static analysis techniques examine the malware as a binary on disk. This has the advantage of viewing the malware in its entirety. The major drawback however is that the malware may be obfuscated, often through the use of software known as packers. Packers encrypt, compress, or otherwise obfuscate the actual malicious code. Once the packed malware is executed, it unpacks itself in order to accomplish its task. Static analysis of a packed malware sample only reveals details about the packer itself, not the underlying malware. Any features extracted from a packed sample are generally not useful for characterizing the malware's behavior [10].

Dynamic analysis techniques require the actual execution of the malware. A malware sample is typically executed in a controlled environment, either emulated or virtualized, for a set period of time. During this time, the malware is monitored and its actions are recorded. This approach has the advantage of determining exactly what code is executed by the malware and seeing the impact it has on the system. Additionally, some packed malware will unpack itself at runtime and execute the underlying malicious code, enabling a bypass of this protection.

In previous works using features extracted from dynamic analysis [11, 12], the malware is categorized, clustered, or grouped using various machine learning algorithms with Windows API calls and other information as features. Clustering algorithms cluster malware into groups based on features. The downside of clustering algorithms is they do not give insights into how malware are actually grouped together because features are usually hard to understand or too large to comprehend by humans. Another common way to group malware is to use family labels provided by VirusTotal [13]. However, the labels are usually inconsistent across multiple providers [14]. In addition, the labels from these providers do not necessarily reflect malware behaviors. For example, labels could be based on the country of origin or packer used to pack each malware. Categorizing malware based on behaviors provides perspective of the overall behaviors of each group of malware. Since we focus on the actual behaviors of malware, we analyze dynamic API call logs collected by running malware samples.

Topic modeling has been used in a number of studies in the area of malware detection and classification. The following studies applied topic modeling to the feature extraction process for API call logs. Sundarkumar et al. [4] proposed a malware detection method applying LDA topics as features for their classification models. A semi-supervised malware classification method was proposed by Kolosnjaji et al. [9]. By joining the results from static and dynamic analyses, they improved the malware classification performance. In their study, topic modeling was used as part of the dynamic analysis to generate features to classify malware families. A text classification method involving topic modeling and Class Association Rule Mining (CARM) was proposed by Kumar and Ravi [15]. They used topic modeling during the dimensionality reduction process.

To evaluate the performance of their method, malware prediction was performed. Inspired by LDA, Xiao and Stibor [8] proposed a probabilistic topic model to reveal behavior patterns and perform malware classification.

2.2. Topic Modeling and Topic Evaluation

Latent semantics in unstructured documents have been explored with various techniques. Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Analysis (LDA) are the major topic modeling techniques. All of them share the same three basic assumptions [5, 16, 17]: (1) there exists latent semantic structure, or topics, in documents; (2) word-document co-occurrences are able to infer topics; and (3) words are related to topics and topics are related to documents. The difference between these techniques are the underlying mathematical frameworks.

LSA applies singular value decomposition (SVD) to a weighted document-term matrix to generate a low-dimensional representation of words and documents [16]. This representation can be further used to find the similarity of words or documents. PLSA uses a probabilistic method instead of SVD [17]. Each document can be represented as a mixture of word-generating topics. Expectation Maximization (EM) is used to fit each PLSA model. Similarly, LDA also represents documents as mixtures of word-generating topics. However, LDA uses a generative model for document-topic mixtures using a Dirichlet prior on a documents' topic distribution. LDA assumes topics exist in a Dirichlet-distributed latent space from which document multinomial topic mixtures are drawn [5].

In general, topic evaluation is performed by calculating various quality measures such as log-perplexity [5] and topic coherence measures [18, 19, 20]. Indirect evaluation can also be done by combining topic models with classifiers [2, 21]. For example, generated topics can be used as features in a classification model. Then, the classification model can be assessed using ground truths.

While the automated quality measures indicate how well each topic model fits to data, they do not necessarily reflect how well humans can interpret the inferred topics. Measuring human interpretability is a challenging task as automated quality measures of topics do not necessarily agree with actual semantic understanding [6, 7]. Evaluations of human interpretability have been done mainly with natural language corpora [6, 7, 3]. Chang et al. [6] proposed two methods for judging human interpretability of topic models: word intrusion and topic intrusion. These methods quantitatively measure how well each topic model generates semantically coherent topics and representative document topic mixtures based on human evaluators.

3. Methods

This work focuses on Windows API calls that were recorded during dynamic analysis of the malware sample. The overall process is shown in Figure 1. The API call logs were collected using Cuckoo Sandbox. The collected API call logs were preprocessed and then fitted using LDA to create topic models at different granularity levels. Word intrusion and topic intrusion tasks used to evaluate human interpretability of topic models were generated for each model and presented to human evaluators. The performance of these tasks were collected to calculate model precision and topic log odds. These are explained in Section 3.4.

3.1. Data Collection

Cuckoo Sandbox is an automated dynamic analysis sandbox capable of running files, monitoring their behaviors, and recording this information [22]. Cuckoo Sandbox records file creations and modifications, registry modifications, process creation, and API calls. For this study, we focused on Cuckoo Sandbox's ability to capture Windows API calls. Cuckoo Sandbox monitors a malware's API calls by injecting a dynamic link library (DLL) into the process of the malware during execution. Any processes spawned by the initial malware sample will also have a DLL injected into them. A modified version of Cuckoo Sandbox, spender-sandbox, was used due to additional features it contained beyond the standard Cuckoo Sandbox 1.2 [23].

To process malware, five hosts running ESXi 5.5.0 were used. Although the hardware varies slightly between the hosts, two of them have 16 physical cores and three of them have 20 physical cores and all five have 128 Gib of RAM. Each host also has an adaptor connected to an isolated network that the hosts share. Each host contains a cuckoo node, a virtual machine (VM) running CentOS 7 and Cuckoo Sandbox. Each cuckoo node has 64 GiB of RAM and 28 virtual cores. The Cuckoo nodes each have 20 Cuckoo agent VMs within them. All together there are 100 Cuckoo agent VMs managed by the five Cuckoo nodes.

Each cuckoo agent is a Windows 7 32-bit VM that was managed using QEMU 2.5.1. The Windows VMs are a fresh install of Windows 7 32-bit constructed with 512 MB of RAM, 1 CPU core, Adobe Reader 11, and Python 2.7 installed. The Windows firewall and User

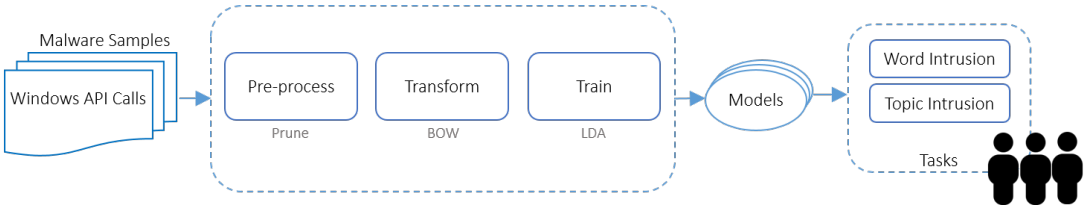

Figure 1. Overall Process

Account Control were disabled.

The cuckoo agent VMs were each connected to an isolated network that did not permit internet access. A VM running INetSim [24] was deployed on this network to simulate various network services (DNS, HTTP, SMTP, etc.). Some malware samples will run additional code upon detecting that these services exist. This was viewed as a reasonable compromise between data collection and ensuring the malware did not cause any harm to the local network or other systems on the internet. If a malware sample did not contact a command and control server to obtain configuration information, it would be unlikely to fully execute [25].

Our malware samples were gathered from VirusShare [26]. From their 2015 to 2016 datasets, we randomly selected 35,177 malware samples. Each sample was analyzed using Cuckoo with a timeout of five minutes and with terminate processes not set. Once a sample finished processing, all recorded information was stored in a report on the results server on the isolated network. Our dataset consisted of Windows API call logs of malware samples extracted from these reports. The statistics regarding API calls are shown in Table 1. Among all malware samples, we discarded 303 malware samples that had less than 20 API calls. We use 20 as the threshold to distinguish inactive malware samples from active ones. Inactive malware samples are those malware binaries that did not perform any actions in Cuckoo Sandbox. After discarding inactive malware samples, our dataset consisted of API call logs from 34,874 malware samples.

Table 1. API Call Log Statistics.

	API Calls
Average	89,851
Maximum	1,327,283
Minimum	2
Median	17,511
Unique	326

3.2. Preprocessing

The obtained API call logs were preprocessed to remove API calls that are not meaningful. Meaningless API calls are those that need to appear with other context (i.e. port numbers and paths) for their meanings to be interpretable. These API calls have vague meanings in themselves and require additional sematic context for their behaviors to be interpretable. Examples of such API call are RegCreateKeyExW and RegQueryValueExA, which require specific registry key to be meaningful. Nearly all software running on a Windows operating system will query the registry for various values, but without knowing the specific key being queried, the meaning is ambiguous.

In addition, any adjacent duplicate API calls were removed and replaced with one API call. For example, NtOpenFile NtReadFile NtReadFile NtReadFile NtCloseFile was converted into NtOpenFile NtReadFile NtCloseFile. We also removed any API calls that appear in less than 10 API call logs (documents). The resulting dataset contained 205 unique tokens and 2,385,248 total tokens.

3.3. Topic Models

Our goal is to establish a baseline of topic interpretability of Windows API calls using LDA. Three different models were fit to compare the interpretability of different granularity levels. Those levels were 20 topics, 30 topics, and 40 topics. Models were fit in bag-of-word transformation using Gensim 2.1.0 [27], which was configured to process with 200 iterations, 5 passes, and 3000 chunksize in multi-processing mode.

3.4. Human Interpretability Tasks

In order to measure human interpretability of topics, participants were asked to perform word intrusion and topic intrusion tasks [6]. Our goal is to quantitatively measure whether humans can find the “intrude” word or topic presented to them. The results were used to calculate two measures for evaluating human interpretability of topic models. This is to evaluate

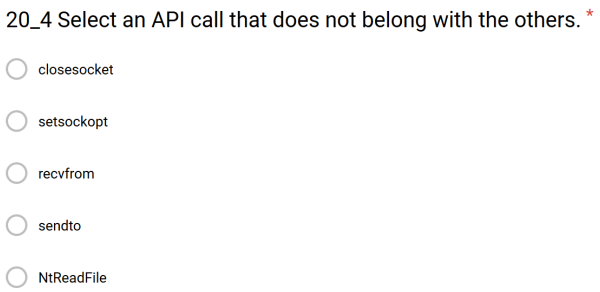

Figure 2. Sample Word Intrusion Question. For this task, the participants were asked to select an API call that does not belong with the rest of the API calls.

whether LDA, which is commonly used to generate topics from natural language document collections, can also be used with API call logs.

3.4.1. Word Intrusion Task To construct a set of word intrusion tasks, we randomly selected ten topics for each model. Then, from each topic, the four most probable words were selected. In addition to these words, an intruder word was selected at random from a pool of words with low probability in the current topic. The intruding word is also required to have high probability in some other topics. All five words were shuffled and presented to the subjects. Figure 2 shows a sample word intrusion question presented to the subjects and Table 2 shows five topics (out of ten) and the intruder presented to the subjects for each topic model.

In the context of API call logs, an API call is equivalent to a word in natural language text and each topic consists of multiple API calls representing a behavioral theme. The model precision was calculated from the results. The model precision is a quantitative measure proposed by Chang et al. [6]. This measure indicates the proportion of correct intruders found by the participants; that is, how well the inferred topics match human concepts.

More formally, let ω_k^m be the index of the intruding word for model m and topic k and $i_{k,s}^m$ be the index of the intruding word selected by subject s, where S represents the number of subjects. The model precision is the fraction of subjects correctly selecting the intruder.

$$\text{MP}_k^m = \frac{1}{S} \sum_s 1(\omega_k^m = i_{k,s}^m) \tag{1}$$

The higher the model precision (MP) the better. The upper bound of this measure is 1 meaning every participant found every intruder correctly for a particular topic model.

3.4.2. Topic Intrusion Task In this task, subjects are shown a snippet from a document, or API call log in our context. Along with the document they are presented with three topics, where each topic is represented by the six highest probability words within that topic. Two of those topics are the highest probability topics assigned to that document. The intruder topic is chosen randomly from the other low-probability topics in the model. We lowered the number of topics and words from the original topic intrusion task proposed by Chang et al. [6] because of the time constraint. We believe that analyzing and finding intruders in API call logs could be more time consuming than analyzing and finding intruding topics in natural language documents.

Similar to the word intrusion task, the subjects were asked to select the topic they think does not agree with the given document. A sample topic intrusion question presented to the subjects are shown in the Figure 5. For this task, the first four API calls in each topics were highlighted in the given API call log presented to the subjects to aid the participants in analyzing each API call log.

Let $\hat{\theta}_d^m$ be the point estimate of the topic proportions vector for document d of model m and $\hat{\theta}_{d,s}^m$ be the intruding topic selected by subject s. In addition, let $\hat{\theta}_{d,*}^m$ be the actual intruding topic and S represents the number of subjects. The topic log odds (TLO) can be calculated as:

$$\text{TLO}_d^m = \frac{1}{S} \sum_s \log \hat{\theta}_{d,*}^m - \log \hat{\theta}_{d,s}^m \tag{2}$$

High topic log odds indicate that the subjects agree with the judgment of the model. The upper bound of this measure is 0 meaning every participant found every intruder correctly for a particular topic model.

3.5. Participants

In order to judge whether the topic model is learning malware behavioral themes from collection of API call logs, we needed to evaluate whether the inferred topics are agreeable and interpretable by human malware analysts. We also sought to compare the automated quality measures to the metrics obtained from the subjects of different models. This result can be used as a baseline to determine which automated measures are appropriate for choosing topic models of API calls. Malware analysts were surveyed on the interpretability of topics from three topic models using the word intrusion and topic intrusion tasks. This study had been reviewed and was granted exemption from IRB

Table 2. Top 4 API calls from 5 topics for each model and random intruder used in the word intrusion task.

20 Topics		
No.	**Top words (API calls)**	**Intruder**
1	LdrLoadDll, Process32NextW, NtClose, NtAllocateVirtualMemory	HttpOpenRequestW
0	NtWaitForSingleObject, NtOpenSection, NtQueryFullAttributesFile, NtClose	NtCreateFile
11	NtClose, NtQueryAttributesFile, NtReadFile, NtQueryInformationFile	NtOpenKey
2	InternetReadFile, InternetCloseHandle, HttpOpenRequestW, InternetConnectW	setsockopt
16	NtClose, NtCreateFile, NtOpenFile, NtReadFile	NtWaitForSingleObject
30 Topics		
No.	**Top words (API calls)**	**Intruder**
26	recv, select, NtClose, send	NtSetInformationFile
12	NtClose, NtDeviceIoControlFile, NtOpenKey, NtQueryValueKey	NtOpenFile
3	NtSetInformationFile, NtReadFile, NtWaitForSingleObject, NtClose	NtOpenKey
20	NtCreateNamedPipeFile, NtOpenFile, NtClose, GetDiskFreeSpaceA	LdrLoadDll
1	LdrLoadDll, NtAllocateVirtualMemory, NtClose, WriteProcessMemory	InternetCloseHandle
40 Topics		
No.	**Top words (API calls)**	**Intruder**
15	NtClose, NtCreateMutant, WriteProcessMemory, NtCreateFile	NtOpenKey
33	CryptCreateHash, CryptHashData, NtClose, NtReadFile	LdrLoadDll
18	Process32NextW, CreateToolhelp32Snapshot, NtClose, Process32FirstW	NtWriteFile
35	CryptCreateHash, CryptHashData, setsockopt, NtClose	NtReadFile
29	InternetReadFile, InternetCloseHandle, HttpOpenRequestW, InternetConnectW	NtClose

approval.

For each model, ten randomly selected topics (behavioral themes) and five randomly selected documents (API call logs) were evaluated by the subjects. The subjects were recruited via email and tasks were performed via Google Forms. Each subject received the same set of instructions on how to complete the tasks. However, the order of word choices and topic choices were shuffled.

3.6. Automated Quality Measures

Two automated metrics were computed to compare model quality and topic coherence across models. The first measure, log-perplexity[5], quantifies how well a probabilistic model predicts a sample. In other words, it is a theoretical measure of the quality of the model for the word prediction task. To calculate log-perplexity, the collection of API call logs was split randomly. We used 80% of API call logs for training and 20% for testing. The training and test sets were only used for computing the log-perplexity for comparison across models. Topic coherence (C_V and C_{UMass}) [18] are two other measures that were computed. They are known to have better performance than log-perplexity in quantifying human interpretability of topics in natural language corpora [18].

4. Results and Discussion

Table 3 contains the results of automated quality measures, log-perplexity, and the topic coherence metrics C_V and C_{UMass}. Lower log-perplexity is desirable, whereas the higher topic coherence measures the better topic interpretability. Based on the log-perplexity computed, the quality of the model increases as the number of topics increases. Similar observations have been made in the existing literature with natural language corpora [3, 6]. However, when we look at topic coherence measures, which are common automated measures of topic interpretability, the results concur with the findings in the existing studies [6, 18]. In each of those studies, the trends of topic coherence measures do not correspond to the trend of log-perplexity. Based on both topic coherence measures, the 20 topic model has the highest level of interpretability and the 30 topic model has the lowest interpretability.

Table 3. Log–perplexity and topic coherence measures for different LDA models fit to API call logs collection.

Topics	**Log–Perplexity**	C_V	C_{UMass}
20	1.539	0.557	-0.376
30	1.534	0.543	-0.410
40	1.532	0.553	-0.389

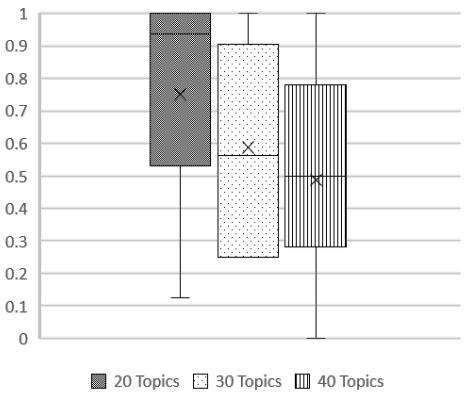

Figure 3. Model Precision

For the word intrusion and topic intrusion tasks, there were 8 participants; each with domain knowledge in Windows malware analysis. Among the participants, 37.5% of them have greater than two years of experience in this specific area and the rest have two years of experience or less.

Table 4. Model precision and topic log odds for different LDA models fit to API call logs collection.

Topics	Model Precision	Topic Log Odds
20	0.750	-0.124
30	0.588	-0.115
40	0.488	-0.082

Table 4 shows model precision and topic log odds results of each model. For the word intrusion task, the participants performed best in the 20 topic model. The mean and median of model precision of each topic model are shown in Figure 3, where a horizontal line indicates a median and an x indicates a mean of model precision.

In the 20 topic model precision, the median and mean are as high as 0.94 and 0.75, respectively. This indicates that the inferred topics in the 20 topic model are highly coherent from the participants' point of view. However, we recognized a substantial difference between these two values, where the mean was affected by the missed responses.

In the 30 topic model, the model precision is 0.59 with the median and mean of 0.56 and 0.59, respectively, suggesting that inferred topics of 30 topic model are moderately coherent.

The performance of participants on word intrusion task for 40 topic model is the lowest among all models. Its model precision of 0.49 (both mean and median are around 0.5) showing that the proportions of correct intruder found and missed are about the same. When comparing the model precision and topic coherence measures across all models, all of them agree than the 20 topic model has the highest coherent topics among all models.

When we looked at topics of the 20 topic model, we found that most of them have a concise meaning that each topic contains one sole behavioral theme. An example of such a topic is a topic representing file creation and manipulation behaviors (NtQueryInformationFile, NtSetInformationFile, NtClose, NtCreateFile) with the intruding API call being NtWaitForSingleObject. All participants correctly located the intruder for this particular topic. Another example of a topic with one specific behavior is socket communication (closesocket, setsockopt, recvfrom, sendto). For this topic, every participant also selected the correct intruder (NtReadFile).

However, when inferred topics contain disparate or mutiple behaviors, it is difficult for human to distinguish the behavioral theme(s). An example of such behavioral themes is object handling and file access (NtWaitForSingleObject, NtOpenSection, NtQueryFullAttributeFile, NtClose), where "object" in this context may not strictly represent a file. For this topic, the true intruder is NtCreateFile, which could be viewed as a non-intruder from the evaluators' standpoint since it is perceived as being related to files. We noticed that in the 30 and 40 topic models, there are more topics in which each topic contains more than one precise behavioral theme compared to the 20 topic model. This caused the performance of word intrusion task of these two models to be lower than the 20 topic model.

Similar to natural language topic models, higher granularity topics could be desirable for human to interpret and to judge the quality of each topic. For example, a topic for "reptiles" is more granular than a topic for "animals". From this example, the first topic is less vague when it comes to judging the detecting an intruder because the intruder will be likely to stand out more when considering it along with other words. In API call topic models, an example of a higher granularity topic is a topic specifically relating to file manipulation alone (i.e. NtReadFile, NtWriteFile, NtQueryInformationFile, NtClose) as opposed to a topic that involves file tranferring and manipulation (i.e. NtClose, NtWriteFile, InternetConnectA, InternetReadFile) because there is a higher chance for a true intruder to be related to either sub-topics in the given topic and therefore harder for the subjects to detect a true intruder.

Based on the studies using natural language corpora [3, 6], higher granularity topics could be achieved by

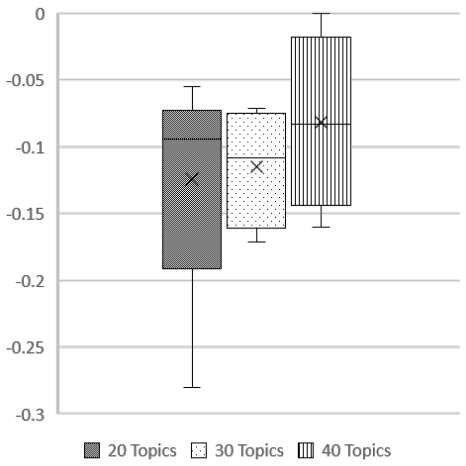

Figure 4. Topic Log Odds

increasing the number of topics. However, we found that the effect is opposite with API call logs, where the proportion of unique tokens per total tokens is much smaller than general unique tokens in natural language corpora (17,993 per 5,820,160 tokens in clinical reports [3], 8,269 per one million tokens in 1987-2007 *New York Times* articles, and 15,273 per three million tokens in a sample of 10,000 articles from *Wikipedia* [6]). We observed that as the number of topics increases, the number of broader behavioral themes also increases making it harder for participants to distinguish the intruding API call.

For the topic intrusion task, the mean and median of topic log odds of each topic model are shown in Figure 4. We observe that the trend of topic log odds deviates from the trend of model precision across models. The 40 topic model achieved the highest mean topic log odds among all three models, where the 20 topic model has the lowest mean topic log odds. Interestingly, the trend of topic log odds across all models is compatible with log-perplexity suggesting that the 40 topic model outperforms the other models in terms of how well the generated behavior groups representing API call logs. However, we noticed that the medians of topic log odds of the 20 topic model and the 40 topic model are very similar (-0.094 and -0.083, respectively) with both of them being higher than the median topic log odds of the 30 topic model. Similar to the model precision result of the 20 topic model, this skew in topic log odds of the 20 topic model caused by missed responses that affected the mean of the results, which is known to be sensitive to outliers.

To examine why the trends of topic log odds and model precision results are not agreeable, we looked into API call logs in which every participant detected intruding topics correctly. We found that each non-intruding topic does not necessarily contains one sole behavioral theme. Figure 5 shows an example of a document and topics presented to the subjects as part of the topic intrusion task. The first non-intruding topic involves driver loading and object handling with "object" being a registry key. The other non-intruding topic also consists of a combination of more than one specific behavioral theme. The participants could still indicate that the given API call log does not exhibit file transmission, which is the intruding topic.

Although the results presented suggest that topic models can learn interpretable API call logs from a malware analysts' perspective, there is much work to be done to improve the rigor of the finding because of our small sample size. With the ultimate goal being to apply topic modeling into an automatic malware behavior categorizing system, the tradeoffs between coherent topics and the ability of topics to represent API call logs should be determined because the trends of model precision and topic log odds results are not compatible. Moreover, with high growth rate of new malware and numerous possibilities of behavioral themes regarding what malware can do, it is challenging to address the scalability of the system to improve the results in the future.

5. Limitations

Due to the relative rarity of human subjects with domain knowledge in Windows malware analysis, it was difficult to find people with the requisite skills to survey for the human interpretability portion of this work. Participants should to be able to distinguish the meanings of API calls and know which ones are related the same way a general group of people should be able to distinguish words in natural language text. They also need to be able to judge whether the given documents or API call logs exhibits the given topics or behavioral themes. This led to a small sample size for the human interpretability survey.

In order to study a large set of malware samples, it is necessary to use an automated solution. Unfortunately purely automated solutions may elicit limited output from a given malware sample. This can be due to a number of reasons. Some malware will examine the system it is running in to determine if it is being monitored. While our system was hardened to prevent this to a degree, it is typically not possible to make an automated, virtualized, analysis environment completely undetectable. If malware determines it is running inside a sandbox environment, it will often simply exit or at least not demonstrate its full behavior.

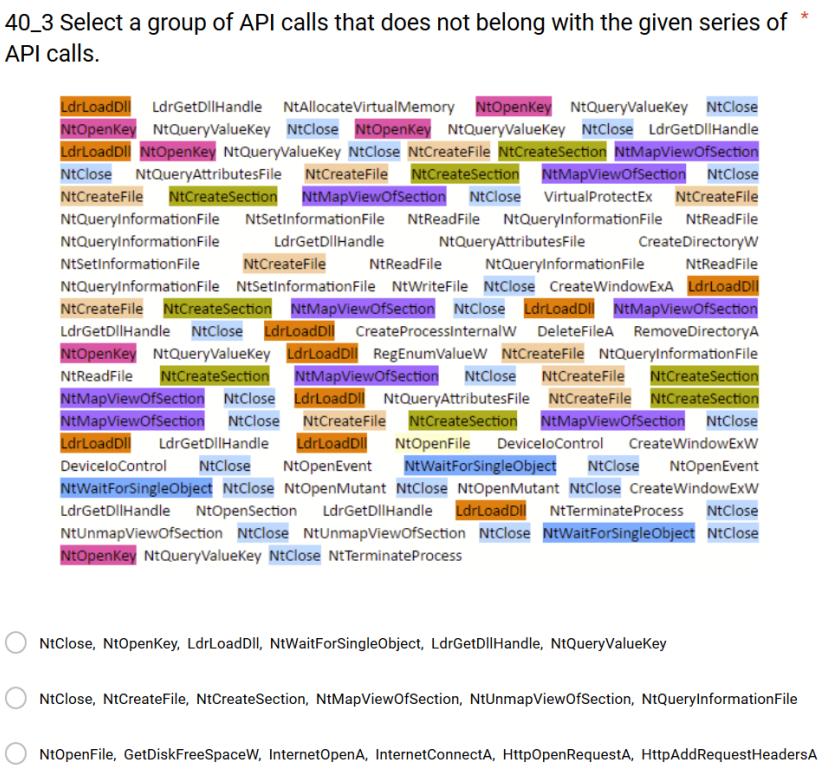

Figure 5. Sample Topic Intrusion Question. For this task, the participants were asked to select a group of API calls that does not represent the behaviors exhibited in the given API call log.

Some malware requires data from the Internet in order to fully operate. This may be an automated process where it downloads a set of waiting commands that it will execute, or it may be a more manual process where the malware needs a human driving it in order to perform various actions such as in the case of most remote access trojans (RATs). Each of these scenarios results in the malware not exhibiting its full capabilities.

6. Conclusion and Future Work

In this work, we have presented an exploratory study to evaluate the interpretability of behavior themes or topics and relevance of topic models learned from API call logs using human ratings. The results were obtained using a general topic model without any modifications for API call logs or additional information from malware analysis. The model precision results are not compatible with log-perplexity, which concur the findings of topic evaluation literature on natural language corpora. However, model precision corresponds topic coherence measures in terms of the highest interpretable topic model. It is observed that the trend of topic log odds does not agree with the trend of model precision across models.

An additional study with a larger human sample population and larger malware dataset to improve the rigor of this finding is planned. The results also encourage further investigation on customized topic models to API call logs to improve the human interpretability. Examining topic models with API calls with other vector space models and different transformations other than bag-of-word is another possibility. The ability to include new topics constantly when new malware samples are added should also be explored.

7. Acknowledgements

The authors would like to thank Distributed Analytics and Security Institute for the financial support. The findings and opinions in this paper belong solely to the authors and does not necessarily represent the official views of the institute.

References

[1] "Malware Statistics." https://www.av-test.org/en/statistics/malware/. Accessed: 2019-09-01.

[2] L. Hong and B. D. Davison, "Empirical Study of

Topic Modeling in Twitter," in *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, (New York, NY, USA), pp. 80–88, ACM, 2010.

[3] C. W. Arnold, A. Oh, S. Chen, and W. Speier, "Evaluating Topic Model Interpretability from a Primary Care Physician Perspective," *Computer Methods and Programs in Biomedicine*, vol. 124, pp. 67–75, Oct. 2016.

[4] G. G. Sundarkumar, V. Ravi, I. Nwogu, and V. Govindaraju, "Malware detection via API calls, topic models and machine learning," in *IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1212–1217, Aug. 2015.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[6] J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei, "Reading Tea Leaves: How Humans Interpret Topic Models," *Neural Information Processing Systems*, vol. 32, pp. 288–296, Jan. 2009.

[7] J. Han Lau, D. Newman, and T. Baldwin, "Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality," *14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 530–539, Jan. 2014.

[8] H. Xiao and T. Stibor, "A Supervised Topic Transition Model for Detecting Malicious System Call Sequences," in *Proceedings of the 2011 Workshop on Knowledge Discovery, Modeling and Simulation*, KDMS '11, (New York, NY, USA), pp. 23–30, ACM, 2011.

[9] B. Kolosnjaji, A. Zarras, T. Lengyel, G. Webster, and C. Eckert, "Adaptive Semantics-Aware Malware Classification," in *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA 2016, (Berlin, Heidelberg), pp. 419–439, Springer-Verlag, 2016.

[10] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee, "Polyunpack: Automating the hidden-code extraction of unpack-executing malware," in *22nd Annual Computer Security Applications Conference (ACSAC)*, pp. 289–300, IEEE, 2006.

[11] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, Behavior-based Malware Clustering," in *The Network and Distributed System Security Symposium (NDSS)*, vol. 9, pp. 8–11, Citeseer, 2009.

[12] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[13] "VirusTotal." `https://www.virustotal.com`. Accessed: 2019-09-01.

[14] J. Canto, M. Dacier, E. Kirda, and C. Leita, "Large Scale Malware Collection : Lessons Learned," in *27th International Symposium on Reliable Distributed Systems (SRDS)*, (Napoli, ITALY), Oct. 2008.

[15] B. S. Kumar and V. Ravi, "A Hybrid Approach Using Topic Modeling and Class-Association Rule Mining for Text Classification: the Case of Malware Detection," in *17th IEEE International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, pp. 261–268, 2018.

[16] S. T. Dumais, "Latent semantic analysis," *Annual Review of Information Science and Technology*, vol. 38, no. 1, pp. 188–230, 2004.

[17] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, pp. 177–196, Jan 2001.

[18] M. Röder, A. Both, and A. Hinneburg, "Exploring the Space of Topic Coherence Measures," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, (New York, NY, USA), pp. 399–408, ACM, 2015.

[19] K. Stevens, P. Kegelmeyer, D. Andrzejewski, and D. Buttler, "Exploring Topic Coherence over Many Models and Many Topics," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, (Stroudsburg, PA, USA), pp. 952–961, Association for Computational Linguistics, 2012.

[20] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin, "Automatic Evaluation of Topic Coherence," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, (Stroudsburg, PA, USA), pp. 100–108, Association for Computational Linguistics, 2010.

[21] M. Pavlinek and V. Podgorelec, "Text Classification Method Based on Self-training and LDA Topic Models," *Expert System with Applications*, vol. 80, no. C, pp. 83–93, 2017.

[22] C. Guarnieri, "Cuckoo Sandbox – Automated Malware Analysis." `https://cuckoosandbox.org/`. Accessed: 2019-09-01.

[23] B. Spengler, "Spender-Sandbox." `https://github.com/spender-sandbox/`. Accessed: 2019-09-01.

[24] T. Hungenberg and M. Eckert, "INetSim." `http://www.inetsim.org/`. Accessed: 2019-09-01.

[25] C. Miller, D. Glendowne, H. Cook, D. Thomas, C. Lanclos, and P. Pape, "Insights gained from constructing a large scale dynamic analysis platform," *Digital Investigation*, vol. 22, pp. S48–S56, 2017.

[26] "Virusshare.com." `https://virusshare.com/`. Accessed: 2019-09-01.

[27] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. `http://is.muni.cz/publication/884893/en`.