UNSUPERVISED ATTACK PATTERN DETECTION IN HONEYPOT DATA USING BAYESIAN TOPIC MODELLING

BY FRANCESCO SANNA PASSINO[1,a], ANASTASIA MANTZIOU[2], DANIYAR GHANI[1], PHILIP THIEDE[1], ROSS BEVINGTON[3], AND NICHOLAS A. HEARD[1]

[1]*Department of Mathematics, Imperial College London, London (United Kingdom),* [a]*f.sannapassino@imperial.ac.uk*

[2]*The Alan Turing Institute, London (United Kingdom)*

[3]*Microsoft Threat Intelligence Center (MSTIC), Cheltenham (United Kingdom)*

Cyber-systems are under near-constant threat from intrusion attempts. Attacks types vary, but each attempt typically has a specific underlying intent, and the perpetrators are typically groups of individuals with similar objectives. Clustering attacks appearing to share a common intent is very valuable to threat-hunting experts. This article explores topic models for clustering terminal session commands collected from honeypots, which are special network hosts designed to entice malicious attackers. The main practical implications of clustering the sessions are two-fold: finding similar groups of attacks, and identifying outliers. A range of statistical topic models are considered, adapted to the structures of command-line syntax. In particular, concepts of primary and secondary topics, and then session-level and command-level topics, are introduced into the models to improve interpretability. The proposed methods are further extended in a Bayesian nonparametric fashion to allow unboundedness in the vocabulary size and the number of latent intents. The methods are shown to discover an unusual MIRAI variant which attempts to take over existing cryptocurrency coin-mining infrastructure, not detected by traditional topic-modelling approaches.

1. Introduction. The increasing reliance of enterprises on information technologies, such as cloud services, gives rise to new challenges for protecting customer data and computer systems from intrusions. To tackle these cyber threats, enterprises increasingly resort to quantitative methods for the development of the next-generation intrusion detection techniques. *Honeypots* play an important role in the detection and understanding of attacker behaviours. A honeypot is a host located within a computer network designed to entice malicious attackers. Security teams use the commands issued by attackers during interactive sessions with the honeypot, as well as other meta-data such as the source IP address, in order to understand the attack and the attacker's intent to better protect their networks from compromise. Honeypots therefore provide cyber analysts with *session data*, where each session is comprised of multiple commands issued by the user; each command can be interpreted as a sequence of *instructions* in *command language* (for example, *shell* programming languages), similar to *words* in natural language.

Honeypot session data provide a rare insight into the operational techniques of cyber attackers, such as their automated or interactive nature, the individual scripting styles and their overall objectives. This makes honeypot tracking systems particularly attractive for developing robust quantitative methods for cyber-security (Highnam et al., 2021). The volume of traffic passing through a honeypot can be surprisingly high, and so automating the understanding of these sessions, classifying them and detecting new emerging patterns provides a challenging research problem which is addressed in this article.

Keywords and phrases: model-based clustering, statistical cyber-security, topic modelling.

Typically, attackers have one main objective after gaining access to a network host. For example, an intruder might want to infect the machine with ransomware, build a cryptocurrency miner, take over existing infrastructure, copy information for data leakage or sale, or collect intel about the organisation. Therefore, each observed session could be thought to have an underlying *latent intent*. Importantly, such intents evolve and change over time, creating new threats for the security of cyber-systems. From a statistical perspective, the problem of estimating latent intents from a collection of attempted attacks can be framed as a *clustering* task. Hence, the main objective of this work is to develop clustering models for command line data observed in cyber-security applications. Such clustering models could then be used for automated online classification of network intrusions, providing a valuable tool for threat experts and enterprises to discover underlying patterns that would have not been easily detectable otherwise. Automated threat detection can be viewed as complementary to deterministic classification frameworks for enterprise attacks (for example, MITRE ATT&CK[1]), providing a further level of sophistication to attack pattern detection.

The analysis of attacker behaviour from command logs has been mainly studied from a machine learning perspective in the literature (Shrivastava, Bashir and Hota, 2019; Crespi et al., 2021; Sadique and Sengupta, 2021). In the present work, ideas borrowed from the literature on topic modelling in text analysis are used to to detect attack patterns, with sessions playing the role of documents and commands playing the role of sentences. Command line instructions are modelled under a *bag-of-words* assumption, leading to a generative model for the instructions dependent on the latent intent characterising the corresponding session. This fundamentally differentiates our approach from mixed membership strategies to language modelling, such as Latent Dirichlet Allocation (LDA, Blei, Ng and Jordan, 2003). The drawbacks of LDA for the scope of modelling command lines are threefold: (i) attackers usually have mainly *one* intent per session; (ii) interpreting the results of a mixed-membership model for attack pattern detection is complex for analysts and threat experts; (iii) models based on LDA often present unidentifiability and convergence difficulties, making reproducibility of results problematic. Such difficulties are addressed in this work, presenting an approach that assigns a *single topic*, or intent, to each session, providing a single label to threat experts which is then easy to interpret through statistical summaries of sessions assigned to the same group. Furthermore, one class of proposed models incorporate the additional idea of command-level intents, establishing a two-level clustering structure. This approach appears to alleviate the convergence issues observed with LDA on session data.

Later models and inferential procedures discussed in this work admit the possibility of an unknown and unbounded number of latent intents and an unbounded vocabulary size. These extensions are particularly important in computer network security, since attack vectors frequently evolve meaning new intents to arise, and new command line instructions will appear. The number of topics in LDA models is usually chosen using scree-plot criteria using the perplexities calculated from a holdout dataset (Teh et al., 2006). However, optimising for perplexity might not yield interpretable topics (Ding, Nallapati and Xiang, 2018). In this work, an alternative strategy based on Bayesian hierarchical nonparametric Griffiths-Engen-McCloskey priors (GEM, Pitman, 2006) is used, admitting the possibility of previously unobserved intents and instructions.

The rest of the article is structured as follows: in the remainder of this section, the data sources used in this work are described along with a review of the related literature. Section 2 describes models for session data, and Section 3 presents inferential procedures. The methodology is then extended to the cases of unbounded numbers of topics and vocabulary size in Section 4. Finally, the proposed methods are applied to real-world session data from honeypots in Section 5, and the practical implications of the results are discussed.

[1] For more details, see https://attack.mitre.org/.

1.1. *Honeypot session data.* When a user connects to a honeypot through certain protocols, a *session* starts, and every action the user performs on this host is recorded until logout, when the session ends. A user will run a sequence of *commands*, which are strings of code which perform actions on the host. Each command comprises a sequence of *words* drawn from the syntax of the chosen protocol. In the following example session, the intruder first attempts to access a convenient directory (through multiples uses of the `cd` command), then tries three methods of downloading a *bash* script from the web (`wget`, `curl` and `tftp get`, representing different commands having the same underlying intention), before attempting to execute and delete the script. The real IP address which was used in the attack is masked using the string `abc.def.ghi.jkl`.

```
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /
wget http://abc.def.ghi.jkl/Zerow.sh
curl -O http://abc.def.ghi.jkl/Zerow.sh
chmod 777 Zerow.sh
sh Zerow.sh
tftp abc.def.ghi.jkl -c get tZerow.sh
chmod 777 tZerow.sh
sh tZerow.sh
rm -rf Zerow.sh tZerow.sh
```

Note that transforming commands into sequences of words, known as *tokenisation* in the literature, is not a trivial task in cyber-security. For example, consider the web address `http://abc.def.ghi.jkl/Zerow.sh`, which appeared in the second command of the session. One might consider the *entire string* as a word, or split it into different words, such as `http`, `abc.def.ghi.jkl` and `Zerow.sh`. Furthermore, the entire Internet Protocol (IP) address `abc.def.ghi.jkl` could be considered as a word, or just its subnet `abc.def`, for example. Similarly, `Zerow` could be considered as an individual word, excluding the file extension `.sh`. More details around the preprocessing of session data will be given in Section 5.1.

1.2. *Related literature.* In topic models, each document is usually considered as a bag-of-words, and the words are assumed to be exchangeable. Under this assumption, the information carried by paragraphs and sentences in natural language is lost. In cyber-security, documents correspond to sessions and sentences correspond to commands, which are expected to have a specific intent. For attack pattern detection, it would be informative to also capture such latent intents at the command-level. In the literature, document and sentence clustering have been considered as two independent problems.

The problem of clustering documents has been extensively studied in the natural language processing, computer science and information retrieval communities (for a survey, see Aggarwal and Zhai, 2012, and references therein). Common approaches include matrix factorisation techniques (Xu, Liu and Gong, 2003) and spectral clustering (Cai, He and Han, 2011). Furthermore, Wallach (2008) proposed a cluster-based topic model extending LDA, where each group is assigned a cluster-specific Dirichlet prior on the document-specific topic distribution. Xie and Xing (2013) also propose a multi-grain topic model with clustering where documents are assigned global and group-specific topics.

Sentence-level structure within topic models has been largely overlooked in the literature. Balikas, Amini and Clausel (2016) propose to extend LDA by sampling words from sentence-specific topic distributions. Furthermore, Jiang et al. (2019) propose to model the sentence-specific topic distribution as a mixture between the topic distributions of adjacent sentences, weighted by a topic association matrix. In the present article, a new framework is proposed which permits to joint inference of the latent structure for both sessions and commands.

Usually, one of the main difficulties for LDA models is topic interpretability. Sparse topic models (Williamson et al., 2010; Archambeau, Lakshminarayanan and Bouchard, 2015; Zhang, 2020) alleviate this issue by enforcing sparsity in the topic-specific word distributions. Doshi-Velez, Wallace and Adams (2015) proposed Graph-Sparse LDA, that used relationships between words to improve interpretability. Also, the performance of LDA methods heavily relies on suitable preprocessing of the data. For example, high-frequency words are often removed, under the assumption that such words make limited contributions to the meaning of the documents. In order to avoid data pruning, alternative term-weighting schemes have been proposed in the literature (Wilson and Chew, 2010). Here a further possible solution is proposed: a *secondary* topic, shared across all documents, can be used to capture high-frequency words and lead to more interpretable *primary* topics characterising individual documents.

Another possible explanation of the issues of LDA with high-frequency terms is that, in natural language, word counts have a power-law distribution (Sato and Nakagawa, 2010). Therefore, Sato and Nakagawa (2010) proposed a Pitman-Yor LDA model, which admits power-laws by construction. Another approach to the problem of modelling power-laws is the latent IBP compound Dirichlet Allocation model (Archambeau, Lakshminarayanan and Bouchard, 2015). It is unclear whether power-laws apply to the word counts in command line data and cyber-security applications. Such structures could be easily accounted for in the methodology proposed in this paper via two-parameter GEM prior distributions, corresponding to stick-breaking proportions of a Pitman-Yor process (Pitman, 2006).

Cyber-security applications require the number of latent intents and vocabulary to be unbounded for practical deployment. Hierarchical Dirichlet Processes (Teh et al., 2006) have been successfully used within the context of LDA models to admit an unbounded number of topics. Furthermore, Zhai and Boyd-Graber (2013) developed an online LDA algorithm with unbounded vocabulary size, proposing a multinomial and n-gram prior distribution for a conventional character language. However, n-grams tend to suffer from data sparsity issues (Allison, Guthrie and Guthrie, 2006). In this work, the words are simply interpreted as tokens, and therefore a GEM prior is employed instead, corresponding to a prior distribution over the natural numbers. This strategy avoids the difficulty of specifying a prior distribution on the command line syntax, which would be an undesirable additional task. Furthermore, GEM priors are also assigned to the number of latent topics. It must be remarked that the task of estimating the number of components in a finite mixture model presents issues with consistency both under finite parametric (Cai, Campbell and Broderick, 2021) and nonparametric priors (Miller and Harrison, 2013, 2014) under model misspecification, unless a prior distribution on the concentration parameter of the Dirichlet process is appropriately specified (Ascolani et al., 2022). Therefore, in practice, it is not expected to *always* recover the exact number in the data generating process.

In cyber-security, command line data have been mainly analysed using two different approaches. The first class of studies uses machine learning techniques to understand attacker behaviour from command logs. Notably, Sadique and Sengupta (2021) aim to predict the next command of the attacker by using an edit distance training model on the sequence of commands input. In a similar setup, Crespi et al. (2021) aim to identify attacker behaviours from command logs using supervised NLP methods. Lastly, Shrivastava, Bashir and Hota (2019) focus on classifying types of attacks from commands using a series of machine learning techniques. The second class of approaches analyses attacker behaviour from session data using Hidden Markov Models, as seen in the studies of Rade et al. (2018) and Deshmukh, Rade and Kazi (2019). However, none of the aforementioned studies consider topic modelling approaches for the analysis of sessions.

2. Models for clustering session data. Command line data are observed in *sessions*, where each session is divided into *commands*, and each command is composed of different *words* drawn from a vocabulary V. Following the standard LDA model (Blei, Ng and Jordan, 2003), for D observed sessions the number of commands N_d in session d and the number $M_{d,j}$ of words in each command j within that session are assumed to be Poisson distributed:

$$N_d \sim \text{Poisson}(\zeta),\ d = 1, \ldots, D,$$
$$M_{d,j} \sim \text{Poisson}(\omega),\ j = 1, \ldots, N_d,$$

$\zeta, \omega \in \mathbb{R}_+$. The i-th word in the j-th command of the d-th session is denoted $w_{d,j,i}$, which has a corresponding probability mass function $\boldsymbol{\xi}_{d,j,i} \in \mathbb{R}_+^{|V|}$ over the vocabulary V.

The stated aim is to develop clustering algorithms for sessions, where the clusters represent shared *intents* of the intruders, or *groups* of attackers with similar behaviour. To achieve this aim, a range of topic model structures are now considered, establishing shared distributions $\boldsymbol{\xi}_{d,j,i}$ across groups of sessions and commands to identify clusters. In particular, this work focuses on two approaches: (i) *Constrained*: Each session has a primary topic and a global secondary topic; (ii) *Hierarchical*: Each session topic is a distribution on command-level topics, which introduces two layers of latent topics. The two modelling approaches are discussed in detail in the next sections.

2.1. *Constrained topic modelling with primary and secondary topics.* As a most basic approach, each document d could have a latent assignment to one of K possible topics, where each topic $k \in \{1, \ldots, K\}$ is characterised by a probability mass function $\boldsymbol{\phi}_k$ on the vocabulary V. Let t_d denote the topic assignment for document d, and let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$ where λ_k denotes the probability that $t_d = k$. It could then be assumed $\boldsymbol{\xi}_{d,j,i} = \boldsymbol{\phi}_{t_d}$, such that conditional on the session-specific topic t_d, all the words $w_{d,j,i}$ in session d are sampled from the same distribution ϕ_{t_d}. Assuming conjugate Dirichlet prior distributions for the probability distributions $\boldsymbol{\lambda}$ and $\{\boldsymbol{\phi}_k\}$ implies the following model:

$$\begin{aligned}
\boldsymbol{\lambda} &\sim \text{Dirichlet}(\boldsymbol{\gamma}),\\
\boldsymbol{\phi}_k &\sim \text{Dirichlet}(\boldsymbol{\eta}),\ k = 1, 2, \ldots, K,\\
t_d \mid \boldsymbol{\lambda} &\sim \text{Categorical}(\boldsymbol{\lambda}),\ d = 1, \ldots, D,\\
w_{d,j,i} \mid t_d, \{\boldsymbol{\phi}_k\} &\sim \text{Categorical}(\boldsymbol{\phi}_{t_d}),\ i = 1, \ldots, M_{d,j},\ j = 1, \ldots, N_d,
\end{aligned} \tag{1}$$

where $\boldsymbol{\gamma} \in \mathbb{R}_+^K, \boldsymbol{\eta} \in \mathbb{R}_+^{|V|}$. In the cyber-attack context, these topics correspond to different *intents*.

The simple model above can be used as a starting point for exploring more complex clustering structures. For example, to better identify differences between topic-specific distributions, it could be assumed that words in a document are sampled either from a topic-specific probability distribution, or from a baseline probability distribution shared across all documents. The shared distribution represents words that are commonly used in *all* sessions, but are not key for characterising the intent of a session. For example, in natural language, such a baseline distribution could give probability mass to conjunctions (e.g. *but*, *and*, *if*), articles (e.g. *a*, *an*, *the*), or pronouns (e.g. *she*, *he*, *they*). Similarly, for command lines in a cybersecurity context, the shared distribution might give weight to common bash commands such as `ls` (list contents), `ps` (list the running processes), or `cd` (change directory). The shared distribution will be used to make the session-specific topics more representative of the attacker's intents, excluding commonly used words.

In particular, an extended model assumes each topic has an associated probability $\theta_k \in [0,1],\ k = 1,\dots,K$, representing the mixing proportion between the topic-specific word distribution $\boldsymbol{\phi}_k$ and the shared distribution $\boldsymbol{\phi}_0$. Each word is then sampled with probability θ_{t_d} from $\boldsymbol{\phi}_{t_d}$, or from $\boldsymbol{\phi}_0$ with probability $1-\theta_{t_d}$, implying the following revised model:

$$
\begin{aligned}
\boldsymbol{\phi}_k &\sim \text{Dirichlet}(\boldsymbol{\eta}),\ k = 0,1,2,\dots,K,\\
\theta_k &\sim \text{Beta}(\alpha_k,\alpha_0),\ k = 1,\dots,K,\\
z_{d,j,i} \mid t_d, \{\theta_k\} &\sim \text{Bernoulli}(\theta_{t_d}), i = 1,\dots,M_{d,j},\ j = 1,\dots,N_d,\\
w_{d,j,i} \mid z_{d,j,i}, t_d, \{\boldsymbol{\phi}_k\} &\sim \text{Categorical}(\boldsymbol{\phi}_{t_d z_{d,j,i}}),\ i = 1,\dots,M_{d,j},\ j = 1,\dots,N_d,.
\end{aligned} \tag{2}
$$

This model essentially imposes a sparsity constraint on LDA, by assuming that each document contains only two topics: (i) a *primary* topic t_d, chosen from K primary topics, and (ii) a *secondary* topic shared across *all* documents, denoted "topic 0" for notational convenience.

2.2. *Hierarchical constrained topic modelling with session-level and command-level clustering.* The models in (1) and (2) assume that words in each command within a given session are sampled from the *same* topic-specific distribution, or from a distribution shared across documents. The information about the structure of a session as a sequence of commands is therefore ignored, which might be limiting in practical settings. Instead, it would be reasonable to assume that session-specific intents share similar commands for specific tasks. Such tasks could be interpreted as *command-level intents*, and the distribution of the tasks characterises the *session-level* topic. Let H be the assumed number of command-level topics. It could be assumed that each session-level topic has an associated H-dimensional probability distribution $\boldsymbol{\psi}_k$ across command-level intents, with each command within a given session being assigned a command-specific topic $s_{d,j} \in \{1,\dots,H\}$ sampled from $\boldsymbol{\psi}_{t_d}$. Conditional on $s_{d,j}$, the words in the command are then sampled independently from a $|V|$-dimensional probability distribution $\boldsymbol{\phi}_{s_{d,j}}$, specific to the command-level topic. Therefore, the model in (1) is extended as follows:

$$
\begin{aligned}
\boldsymbol{\psi}_k &\sim \text{Dirichlet}(\boldsymbol{\tau}),\ k = 1,2,\dots,K,\\
\boldsymbol{\phi}_h &\sim \text{Dirichlet}(\boldsymbol{\eta}),\ h = 1,2,\dots,H,\\
s_{d,j} \mid t_d, \{\boldsymbol{\psi}_k\} &\sim \text{Categorical}(\boldsymbol{\psi}_{t_d}),\ j = 1,\dots,N_d,\\
w_{d,j,i} \mid s_{d,j}, \{\boldsymbol{\phi}_h\} &\sim \text{Categorical}(\boldsymbol{\phi}_{s_{d,j}}),\ i = 1,\dots,M_{d,j},\ j = 1,\dots,N_d,
\end{aligned} \tag{3}
$$

where $\boldsymbol{\tau} \in \mathbb{R}_+^H$. In this model, there are two layers of topics, and corresponding indices: (i) Command topic indices, $s_{d,j}$, used to match the words in the corresponding command to distributions $\boldsymbol{\phi}_1,\dots,\boldsymbol{\phi}_H$ over V; (ii) Document topic indices, t_d, used to match the commands in the corresponding session to distributions $\boldsymbol{\psi}_1,\dots,\boldsymbol{\psi}_K$ over the command-level topics. Letting $\boldsymbol{\Phi}$ be the $H \times |V|$ matrix with j-th row $\boldsymbol{\phi}_j$, and letting $\boldsymbol{\Psi}$ be the $K \times H$ matrix with k-th row $\boldsymbol{\psi}_k$, then marginally $\boldsymbol{\xi}_{d,j,i} = \boldsymbol{\lambda}^\top \cdot \boldsymbol{\Psi} \cdot \boldsymbol{\Phi}$, whereas $\boldsymbol{\xi}_{d,j,i} = \boldsymbol{\phi}_{s_{j,d}}$ conditionally.

2.3. *Combining the two approaches: hierarchical constrained topic modelling with secondary topics.* In order to aid interpretability of the command-level topics, it is possible to use the same constraint from model (2). In particular, it could be assumed that words are sampled either from $\boldsymbol{\phi}_{s_{d,j}}$, where $s_{d,j}$ is the command-level topic, or from a distribution $\boldsymbol{\phi}_0$ shared across all commands and sessions. As in (2), each command-level topic $h \in \{1,\dots,H\}$ has an associated mixture probability $\theta_h \in [0,1]$ for sampling words from $\boldsymbol{\phi}_h$. Therefore, the full model, which combines (1), (2) and (3), takes the following form:

$$
\boldsymbol{\lambda} \sim \text{Dirichlet}(\boldsymbol{\gamma}),
$$

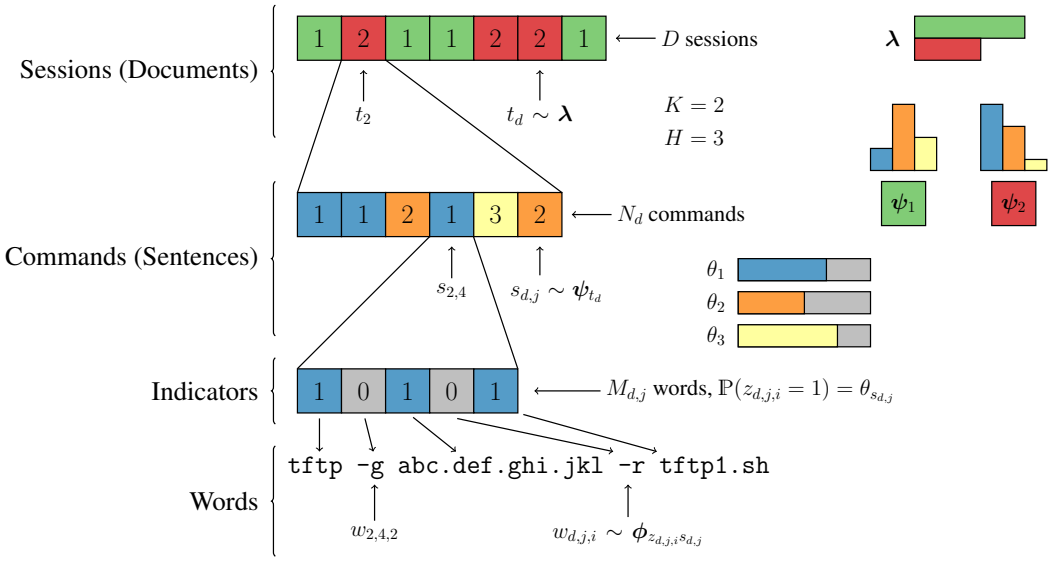


FIG 1: *Cartoon representation of the full Hierarchical Constrained Topic Model (HCTM).*

$$
\begin{aligned}
\boldsymbol{\psi}_k &\sim \text{Dirichlet}(\boldsymbol{\tau}),\ k = 1, 2, \ldots, K, \\
\boldsymbol{\phi}_h &\sim \text{Dirichlet}(\boldsymbol{\eta}),\ h = 0, 1, \ldots, H, \\
\theta_h &\sim \text{Beta}(\alpha_h, \alpha_0),\ h = 1, 2, \ldots, H, \\
N_d &\sim \text{Poisson}(\zeta),\ d = 1, \ldots, D, \\
M_{d,j} &\sim \text{Poisson}(\omega),\ d = 1, \ldots, D,\ j = 1, \ldots, N_d, \\
t_d \mid \boldsymbol{\lambda} &\sim \text{Categorical}(\boldsymbol{\lambda}),\ d = 1, \ldots, D, \\
s_{d,j} \mid t_d, \{\boldsymbol{\psi}_k\} &\sim \text{Categorical}(\boldsymbol{\psi}_{t_d}),\ j = 1, \ldots, N_d, \\
z_{d,j,i} \mid s_{d,j}, \{\theta_h\} &\sim \text{Bernoulli}(\theta_{s_{d,j}}), i = 1, \ldots, M_{d,j}, \\
w_{d,j,i} \mid z_{d,j,i}, s_{d,j}, \{\boldsymbol{\phi}_h\} &\sim \text{Categorical}(\boldsymbol{\phi}_{s_{d,j} z_{d,j,i}}),\ i = 1, \ldots, M_{d,j}.
\end{aligned} \tag{4}
$$

A pictorial representation of model (4) is given in Figure 1. Note that it might be possible to consider variations of (4) through making changes to the specification of the prior distributions on the hyperparameters. For example, document-specific mixing topic proportions $\theta_d \sim \text{Beta}(\alpha, \alpha_0)$ could be used.

The next section will describe inferential methods for model (4). Deriving the inferential procedures for (1), (2) and (3) follows similar guidelines, with minor modifications to the equations used for (4).

3. Bayesian inference via Markov Chain Monte Carlo. This section describes inferential procedures for the topic models discussed in Section 2. The full model is considered, with primary-secondary topics and session-level and command-level clustering. The posterior distribution of the parameters is only available up to a normalising constant, therefore inference must be performed using Markov Chain Monte Carlo (MCMC) methods. The main objective of the inferential procedure is estimating $\boldsymbol{t}$, the session-level clusters. Hence, the remaining parameters could be interpreted as nuisance, and integrated out when possible. The parameters $\boldsymbol{\theta}, \boldsymbol{\lambda}, \{\boldsymbol{\phi}_h\}$ and $\{\boldsymbol{\psi}_k\}$ can be analytically marginalised, resulting in the marginal

posterior density

$$p(\boldsymbol{z}, \boldsymbol{s}, \boldsymbol{t} \mid \boldsymbol{w}) \propto p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{t}) = p(\boldsymbol{t}) \times p(\boldsymbol{s} \mid \boldsymbol{t}) \times p(\boldsymbol{z} \mid \boldsymbol{s}) \times p(\boldsymbol{w} \mid \boldsymbol{z}, \boldsymbol{s}). \tag{5}$$

Each term in the right-hand side of the marginal posterior (5) can be calculated explicitly by conjugacy of the Categorical-Dirichlet and Beta-Bernoulli distributions. The marginal distribution for the session-level intents is:

$$p(\boldsymbol{t}) = \frac{B(\boldsymbol{\gamma} + \boldsymbol{T})}{B(\boldsymbol{\gamma})}, \tag{6}$$

where $B(\boldsymbol{x}) = \prod_i \Gamma(x_i) / \Gamma(\sum_i x_i)$ is the multivariate beta function, and $\boldsymbol{T} = (T_1, \ldots, T_K)$, where $T_k = \sum_d \mathbb{I}_{\{k\}}(t_d)$ denotes the number of sessions assigned to topic k. Similar calculations lead to the marginal distribution for the command-level topics, conditional on the session-level intents:

$$p(\boldsymbol{s} \mid \boldsymbol{t}) = \prod_{k=1}^{K} \frac{B(\boldsymbol{\tau} + \boldsymbol{S}_k)}{B(\boldsymbol{\tau})}, \tag{7}$$

where $\boldsymbol{S}_k = (S_{1,k}, \ldots, S_{H,k})$, and $S_{k,h} = \sum_{d:t_d=k} \sum_j \mathbb{I}_{\{h\}}(s_{d,j})$ denotes the number of commands assigned to the command-level topic h, only from the subset of sessions with session-level topic k. Similarly, the marginal distribution of the primary-secondary topic indicators $\boldsymbol{z}$ has a closed form expression from the Beta-Bernoulli conjugacy:

$$p(\boldsymbol{z} \mid \boldsymbol{s}) = \prod_{h=1}^{H} \frac{B(Z_h + \alpha_h, M_h^* - Z_h + \alpha_0)}{B(\alpha_h, \alpha_0)}, \tag{8}$$

where $Z_h = \sum_{(d,j):s_{d,j}=h} \sum_{i=1}^{M_{d,j}} z_{d,j,i}$ denotes the number of words assigned to the primary topic, only from commands with command-level topic h, and $M_h^* = \sum_{(d,j):s_{d,j}=h} M_{d,j}$ denotes the total number of words in commands with topic h, across all documents. The final component of the marginal posterior (5) is the marginal likelihood for the observed words $\boldsymbol{w}$, conditional on the indicators $\boldsymbol{z}$ and topic-level allocations $\boldsymbol{s}$:

$$p(\boldsymbol{w} \mid \boldsymbol{z}, \boldsymbol{s}) = \prod_{h=0}^{H} \frac{B(\boldsymbol{W}_h + \boldsymbol{\eta})}{B(\boldsymbol{\eta})}, \tag{9}$$

where $\boldsymbol{W}_h = (W_{h,1}, \ldots, W_{h,|V|})$, and $W_{h,v} = \sum_{i,j,d} \mathbb{I}_{\{h\}}(z_{d,j,i} s_{d,j}) \mathbb{I}_{\{v\}}(w_{d,j,i})$ denotes the number of times word v is assigned to the command-level topic h.

The marginal distributions (6), (7), (8) and (9) are the building blocks for the collapsed Gibbs sampler (Liu, 1994) used for inference on the model parameters. The Gibbs sampler consists of three basic moves: resample the session-level topic allocations $\boldsymbol{t}$, resample the command-level topic allocations $\boldsymbol{s}$, and resample the primary-secondary topic indicators $\boldsymbol{z}$. Furthermore, convergence of Gibbs sampling algorithms for clustering usually benefits from split-merge proposals, which are evaluated using a Metropolis-Hastings acceptance ratio, resulting in a collapsed Metropolis-within-Gibbs algorithm. In this framework, split-merge moves can be used on the session-level topics $\boldsymbol{t}$ and command-level topics $\boldsymbol{s}$. The next subsections give a detailed description of the steps required in the Gibbs sampling algorithm.

3.1. *Resampling the session-level topic allocations.* The Gibbs sampler requires to sample from the conditional distribution of a subset of the parameters, conditional on the observed data and remaining parameters. Therefore, for resampling the session-level topic allocation t_d for a given document, it is required to sample from $p(t_d \mid \boldsymbol{t}^{-d}, \boldsymbol{w}, \boldsymbol{z}, \boldsymbol{s})$, where the

superscript $-d$ denotes that the calculations of the corresponding quantity *exclude* the d-th document. For the r-th session-level topic, the probability can be written as:

$$p(t_d = r \mid \boldsymbol{t}^{-d}, \boldsymbol{w}, \boldsymbol{z}, \boldsymbol{s}) \propto p(t_d = r \mid \boldsymbol{t}^{-d}) p(\boldsymbol{s} \mid t_d = r, \boldsymbol{t}^{-d}) \propto \frac{B(\boldsymbol{\gamma} + \boldsymbol{T})}{B(\boldsymbol{\gamma} + \boldsymbol{T}^{-d})} \prod_{k=1}^{K} B(\boldsymbol{\tau} + \boldsymbol{S}_k).$$

where the quantities $\boldsymbol{T}$ and $\boldsymbol{S}_k$ in the final part of the expression are calculated assuming that $t_d = r$. The ratio of beta functions in the conditional distribution can be simplified using the properties of the gamma function, yielding $B(\boldsymbol{\gamma} + \boldsymbol{T})/B(\boldsymbol{\gamma} + \boldsymbol{T}^{-d}) \propto (\gamma_r + T_r^{-d})$. Similarly, the product of beta functions in the final part of the expression could be further simplified using the fact that $S_{r,h} = S_{r,h}^{-d} + S_h^d$, where $S_{k,h}^{-d} = \sum_{u:t_u=k, u\neq d} \sum_j \mathbb{I}_{\{h\}}(s_{j,u})$ and $S_h^d = \sum_j \mathbb{I}_{\{h\}}(s_{d,j})$. From the properties of the gamma function, the probability can be then expressed as:

$$p(t_d = r \mid \boldsymbol{t}^{-d}, \boldsymbol{w}, \boldsymbol{z}, \boldsymbol{s}) \propto (\gamma_r + T_r^{-d}) \frac{\prod_{h=1}^{H} \prod_{\ell=1}^{S_h^d} (\tau_h + S_{r,h}^{-d} + \ell - 1)}{\prod_{\ell=1}^{N_d} \{\sum_{h=1}^{H} (\tau_h + S_{r,h}^{-d}) + \ell - 1\}}. \tag{10}$$

3.2. *Resampling the command-level topic allocations.* The Gibbs sampler also requires to sample the command-level topic allocations from the distribution $p(s_{d,j} = \ell \mid \boldsymbol{s}^{-d,j}, \boldsymbol{w}, \boldsymbol{t}, \boldsymbol{z})$, where the superscript denotes that the quantities have been calculated excluding the (d, j)-th terms. For the ℓ-th command-level topic, the probability can be factorised as:

$$\begin{aligned} p(s_{d,j} = \ell \mid \boldsymbol{s}^{-d,j}, \boldsymbol{w}, \boldsymbol{t}, \boldsymbol{z}) \propto \\ \propto p(s_{d,j} = \ell, \boldsymbol{s}^{-d,j} \mid \boldsymbol{t}) \times p(\boldsymbol{z} \mid s_{d,j} = \ell, \boldsymbol{s}^{-d,j}) \times p(\boldsymbol{w} \mid s_{d,j} = \ell, \boldsymbol{s}^{-d,j}, \boldsymbol{z}). \end{aligned} \tag{11}$$

The first term in the factorisation (11), corresponding to (7), can be simplified noting that $S_{t_d,\ell} = S_{t_d,\ell}^{-d,j} + 1$:

$$p(s_{d,j} = \ell, \boldsymbol{s}^{-d,j} \mid \boldsymbol{t}) \propto (\tau_\ell + S_{t_d,\ell}^{-d,j}). \tag{12}$$

A similar reasoning could be used to simplify the second term in the factorisation (11). In particular, $Z_\ell = Z_\ell^{-d,j} + Z^{d,j}$, where $Z_h^{-d,j} = \sum_{(u,q):s_{u,q}=h,(u,q)\neq(d,j)} \sum_{i=1}^{M_{d,j}} z_{u,q,i}$ and $Z^{d,j} = \sum_{i=1}^{M_{d,j}} z_{d,j,i}$. Using the properties of the gamma function, the corresponding probability is expressed as:

$$\begin{aligned} p(\boldsymbol{z} \mid s_{d,j} = \ell, \boldsymbol{s}^{-d,j}) \propto \\ \propto \frac{\prod_{q=0}^{Z^{d,j}-1} (\alpha_\ell + Z_\ell^{-d,j} + q) \prod_{u=0}^{M_{d,j}-Z^{d,j}-1} (\alpha_0 + M_\ell^{*-d,j} - Z_\ell^{-d,j} + u)}{\prod_{q=0}^{M_{d,j}-1} (\alpha_0 + \alpha_\ell + M_\ell^{*-d,j} + q)}. \end{aligned} \tag{13}$$

The last term in (11) admits a similar simplification to (10), using $W_{\ell,v} = W_{\ell,v}^{-d,j} + W_v^{d,j}$, where $W_{h,v}^{-d,j} = \sum_{u,q,i:z_{u,q,i}s_{u,q}=h,(u,q)\neq(d,j)} \mathbb{I}_{\{v\}}(w_{u,q,i})$, $W_v^{d,j} = \sum_{i:z_{d,j,i}s_{d,j}\neq 0} \mathbb{I}_{\{v\}}(w_{d,j,i})$. The resulting probability is:

$$p(\boldsymbol{w} \mid s_{d,j} = \ell, \boldsymbol{s}^{-d,j}, \boldsymbol{z}) \propto \frac{\prod_{v=1}^{|V|} \prod_{q=1}^{W_v^{d,j}} (\eta_v + W_{\ell,v}^{-d,j} + q - 1)}{\prod_{q=1}^{\sum_v W_v^{d,j}} \{\sum_{v=1}^{|V|} (\eta_v + W_{\ell,v}^{-d,j}) + q - 1\}}. \tag{14}$$

The probability (11) is obtained by calculating the product of the terms (12), (13), (14), and normalising.

3.3. *Resampling the primary-secondary topic indicators.* In the models with primary-secondary topics, the Gibbs sampler also requires to resample the binary indicators $z_{d,j,i}$, conditional on $\boldsymbol{w}, \boldsymbol{s}, \boldsymbol{t}$ and $\boldsymbol{z}^{-d,j,i}$, denoting all the indicators except $z_{d,j,i}$. Each binary indicator is drawn from a Bernoulli distribution with unnormalised probabilities:

$$p(z_{d,j,i} = b \mid \boldsymbol{z}^{-d,j,i}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{t}) \propto p(z_{d,j,i} = b \mid \boldsymbol{z}^{-d,j,i}, \boldsymbol{s}) \times p(\boldsymbol{w} \mid z_{d,j,i} = b, \boldsymbol{z}^{-d,j,i}, \boldsymbol{s}), \tag{15}$$

where $b \in \{0, 1\}$. Noting that $Z_{s_{d,j}} = Z^{-d,j,i}_{s_{d,j}} + b$, where the term $Z^{-d,j,i}_{s_{d,j}}$ is defined as $Z^{-d,j,i}_{h} = \sum_{(u,q):s_{u,q}=h,(u,q)\neq(d,j)} \sum_{i=1}^{M_{u,q}} z_{u,q,i}$, the first term in the factorisation becomes:

$$p(z_{d,j,i} = b \mid \boldsymbol{z}^{-d,j,i}, \boldsymbol{s}) \propto (\alpha_{s_{d,j}} + Z^{-d,j,i}_{s_{d,j}})^{b} (\alpha_0 + M^{*}_{s_{d,j}} - Z^{-d,j,i}_{s_{d,j}} - 1)^{1-b}.$$

For the marginal likelihood of observed words, the only terms affected by a change in the binary indicator $z_{d,j,i}$ are $W_{0,w_{d,j,i}} = W^{-d,j,i}_{0,w_{d,j,i}} + 1 - b$ and $W_{s_{d,j},w_{d,j,i}} = W^{-d,j,i}_{s_{d,j},w_{d,j,i}} + b$, where $W^{-d,j,i}_{h,v} = \sum_{(u,q,r):z_{u,q,r}s_{u,q}=h,(u,q,r)\neq(d,j,i)} \mathbb{I}_{\{v\}}(w_{u,q,r})$, giving:

$$p(\boldsymbol{w} \mid z_{d,j,i} = b, \boldsymbol{z}^{-d,j,i}, \boldsymbol{s}) \propto \left\{ \frac{W^{-d,j,i}_{0,w_{d,j,i}} + \eta_{w_{d,j,i}}}{\sum_{v=1}^{|V|}(W^{-d,j,i}_{0,v} + \eta_v)} \right\}^{1-b} \left\{ \frac{W^{-d,j,i}_{s_{d,j},w_{d,j,i}} + \eta_{w_{d,j,i}}}{\sum_{v=1}^{|V|}(W^{-d,j,i}_{s_{d,j},v} + \eta_v)} \right\}^{b}.$$

3.4. *Split-merge topic allocations.* There are two types of split-merge moves that can be proposed: (i) split-merge session-level topics, and (ii) split-merge command-level topics. For the split-merge move on session-level topics, two sessions d and d' are sampled at random from the D observed sessions. If $t_d = t_{d'} = t^*$, the proposal for the session-level topics splits the sessions assigned to t^* in two different clusters using the following iterative procedure: (i) assign topic t^* to document d, and topic $\tilde{t}$ to document d' (where $\tilde{t}$ corresponds to the number of non-empty clusters, plus one – note that if $\tilde{t} > K$ the split move should be immediately rejected); (ii) documents previously assigned to topic t^* are sequentially allocated to topics t^* or $\tilde{t}$ in random order, with probabilities proportional to the predictive distribution (10), restricted to the session already reallocated to topics t^* and $\tilde{t}$. This allocation procedure is adapted from common split-merge MCMC moves in related clustering problems (see, for example, Dahl, 2003; Sanna Passino and Heard, 2020). The final proposal is denoted as $\boldsymbol{t}^*$, with probability $q(\boldsymbol{t}^* \mid \boldsymbol{t})$, corresponding to the product of sequential probabilities obtained in the splitting procedure. The resulting acceptance probability for the move from $\boldsymbol{t}$ to $\boldsymbol{t}^*$ is:

$$\min\left\{1, \frac{p(\boldsymbol{t}^*)p(\boldsymbol{s} \mid \boldsymbol{t}^*)}{p(\boldsymbol{t})p(\boldsymbol{s} \mid \boldsymbol{t})q(\boldsymbol{t}^* \mid \boldsymbol{t})}\right\}. \tag{16}$$

On the other hand, if $t_d \neq t_{d'}$, the proposal $\boldsymbol{t}^*$ assigns topic t_d to all documents previously given topic $t_{d'}$, corresponding to a merge move. In this case, the acceptance ratio in (16) must be further multiplied by the proposal probability $q(\boldsymbol{t} \mid \boldsymbol{t}^*)$, calculated by simulating a split move from $\boldsymbol{t}^*$ to $\boldsymbol{t}$. Since there is only one way to merge two topics, the proposal probability at the denominator of (16) is $q(\boldsymbol{t}^* \mid \boldsymbol{t}) = 1$ for a merge move.

A similar split-merge move can be constructed for the command-level topics: two commands j and j' are randomly sampled from two random documents d and d' respectively. If $s_{d,j} \neq s_{j',d'}$, a merge move is proposed. Alternatively, if $s_{d,j} = s_{j',d'} = s^*$, the split move proceeds similarly to the procedure described for the topic-level sessions, and the command previously assigned topic s^* are sequentially allocated to s^* or $\tilde{s}$ (corresponding to the number of non-empty command-level topics, plus one) with probabilities proportional to the predictive distribution (11), limited to the commands already reassigned to s^* and $\tilde{s}$. As before, if $\tilde{s} > H$, the move is rejected. In summary, the acceptance probability for a vector of command-level topics $\boldsymbol{s}^*$ obtained via the split-merge procedure is:

$$\min\left\{1, \frac{p(\boldsymbol{s}^* \mid \boldsymbol{t})p(\boldsymbol{z} \mid \boldsymbol{s}^*)p(\boldsymbol{w} \mid \boldsymbol{z}, \boldsymbol{s}^*)q(\boldsymbol{s} \mid \boldsymbol{s}^*)}{p(\boldsymbol{s} \mid \boldsymbol{t})p(\boldsymbol{z} \mid \boldsymbol{s})p(\boldsymbol{w} \mid \boldsymbol{z}, \boldsymbol{s})q(\boldsymbol{s}^* \mid \boldsymbol{s})}\right\},$$

where the probabilities $q(\boldsymbol{s}^* \mid \boldsymbol{s})$ and $q(\boldsymbol{s} \mid \boldsymbol{s}^*)$ are either 1, or the product of allocation probabilities calculated from the sequential splitting procedure.

3.5. *Initialisation schemes.* In MCMC, setting good initial values could be helpful to achieve faster convergence, in particular for complex inferential tasks. In this work, two methods for initialisation are considered, based on spectral clustering and standard LDA.

Spectral methods are commonly used for text analysis and topic modelling (Ke and Wang, 2022). In order to initialise the algorithm via spectral clustering, a $(\sum_{d=1}^{D} N_d) \times |V|$ word occurrence matrix $\boldsymbol{C} = \{C_{sw}\}$ is constructed, where C_{sw} counts the number of times word w appears in command s. Note that all commands are stacked in an individual matrix $\boldsymbol{C}$, initially disregarding information about the division into sessions. A truncated singular value decomposition of $\boldsymbol{C}$ is then calculated, considering only the largest H singular values and corresponding left singular vectors. A clustering algorithm, like k-means, is then run on the resulting embedding, setting H clusters. For initialisation of the session-level topics, a similar procedure is carried out, using the initial values of the command-level topics as words in a spectral clustering algorithm, obtaining a different form of the matrix of counts $\boldsymbol{C}$. First, the matrix $\boldsymbol{C}$, with dimension $D \times H$, is constructed, where each entry C_{dh} counts the number of times a command assigned to the command-level topic h appears in document d. Then, a K-dimensional truncated spectral decomposition of $\boldsymbol{C}$ is calculated, and k-means with K clusters is run on the resulting embedding, obtaining initial values for the session-level topics.

Alternatively, standard LDA could be used to initialise the MCMC sampler, via fast-performing software libraries such as *python*'s *gensim* (Řehůřek and Sojka, 2010). First, LDA with H topics could be fitted, and subsequently used to predict a topic for all the words appearing in commands and sessions. Then the most common estimated topic within each command is selected as the initial command-level topic. If secondary topics are used, LDA is initially fitted with $H + 1$ topics, and the most common estimated topic is selected as secondary topic. The command-level primary topic is then selected as the most common topic within each command, excluding the secondary topic. After command-level topics are estimated, session-level topics could be initialised by running LDA with K topics, using the estimated command-level topics as words within the algorithm. For each command-level topic, now interpreted as a word, a topic can be estimated from the fitted LDA model, and the session-level topics are then initialised as the most common topic within each command.

For initialisation of the primary-secondary topic indicators, $z_{d,j,i}$ could be initially set to 1 if the proportion of sessions or commands where the word $w_{d,j,i}$ appears is less than a pre-specified threshold. This is because $\boldsymbol{\phi}_0$ should represent a distribution of common words, shared across topics.

4. Unbounded number of topics and vocabulary. All models discussed in Section 2 assume a fixed size $|V|$ of the vocabulary, and a fixed number of session-level and command-level topics, K and H respectively. Such assumptions might be problematic if the model is used for clustering future sessions, since it would not be possible to cluster *new* commands, composed of words not present in the vocabulary. Therefore, a potentially *infinite* vocabulary must be considered. Also, the behaviour of attackers is expected to evolve and change over time, and it is possible that *new* attack patterns or intents arise. Therefore, for real-world attack pattern detection, it is beneficial to assume an unbounded number of session-level and command-level topics. These allowances require a modification to the Dirichlet distributions used in Section 2, instead assuming:

$$\boldsymbol{\lambda} \sim \text{GEM}(\gamma), \qquad \boldsymbol{\psi}_k \sim \text{GEM}(\tau),\ k = 0, 1, 2, \ldots, \qquad \boldsymbol{\phi}_\ell \sim \text{GEM}(\eta),\ \ell = 0, 1, 2, \ldots,$$

where $\tau, \eta, \gamma \in \mathbb{R}_+$. The GEM (Griffiths-Engen-McCloskey) distribution (Pitman, 2006) corresponds to the proportions calculated using the stick-breaking representations of the Dirichlet process (Sethuraman, 1994). Hence, the GEM distribution also corresponds to the limit for $K \to \infty$, $H \to \infty$ and $|V| \to \infty$ of the Dirichlet distributions in Section 2 with $\boldsymbol{\gamma} = \gamma \mathbf{1}_K / K$, $\boldsymbol{\tau} = \tau \mathbf{1}_H / H$, and $\boldsymbol{\eta} = \eta \mathbf{1}_{|V|} / |V|$. To simplify the discussion on the GEM distribution, its link to the Dirichlet process, and its representation in the posterior distribution, consider n objects allocated to K_n non-empty groups, with labels $\boldsymbol{x}_n = (x_1, \ldots, x_n)$, such that $x_i \in \mathbb{N}$ and $\max(\boldsymbol{x}_n) = K_n$. Under a Dirichlet process with scaling parameter β, the predictive distribution for the next label in the sequence is:

$$p(x_{n+1} \mid \boldsymbol{x}_n) = \frac{\beta}{\beta + n} \mathbb{I}_{\{K_n+1\}}(x_{n+1}) + \sum_{k=1}^{K_n} \frac{N_{kn}}{\beta + n} \mathbb{I}_{\{k\}}(x_{n+1}), \tag{17}$$

where $N_{kn} = \sum_{i=1}^{n} \mathbb{I}_{\{k\}}(x_i)$ is the number of the n objects allocated to group k. The predictive equation (17) immediately provides a technique for Gibbs sampling: since the Dirichlet process assumes exchangeability of observations, any label can be considered as the last element of the sequence, and a new value resampled using (17). This fact will be particularly useful when implementing the sampler. Using (17), the joint distribution for the sequence is:

$$p(\boldsymbol{x}_n) = \prod_{j=1}^{n} p(x_j \mid \boldsymbol{x}_{j-1}) = \frac{\alpha^{K_n} \Gamma(\alpha)}{\Gamma(\alpha + n)} \prod_{k=1}^{K_n} \Gamma(N_{kn}).$$

It follows that the components of the marginalised posterior distribution (5) take the following revised form:

$$p(\boldsymbol{t}) = \frac{\gamma^{K(\boldsymbol{t})} \Gamma(\gamma)}{\Gamma(\gamma + D)} \prod_{k=1}^{K(\boldsymbol{t})} \Gamma(T_k), \tag{18}$$

$$p(\boldsymbol{s} \mid \boldsymbol{t}) = \prod_{k=1}^{K(\boldsymbol{t})} \frac{\tau^{\sum_{h=1}^{H(\boldsymbol{s})} \mathbb{I}_{\mathbb{N}_+}(S_{k,h})} \Gamma(\tau)}{\Gamma(\tau + \sum_{d:t_d=k} N_d)} \prod_{h:S_{k,h}>0} \Gamma(S_{k,h}),$$

$$p(\boldsymbol{z} \mid \boldsymbol{s}) = \prod_{h=1}^{H(\boldsymbol{s})} \frac{B(Z_h + \alpha_h, M_h^* - Z_h + \alpha_0)}{B(\alpha_h, \alpha_0)},$$

$$p(\boldsymbol{w} \mid \boldsymbol{z}, \boldsymbol{s}) = \prod_{h=0}^{H(\boldsymbol{s})} \frac{\eta^{\sum_{v=1}^{V(\boldsymbol{w})} \mathbb{I}_{\mathbb{N}_+}(W_{h,v})} \Gamma(\eta)}{\Gamma(\eta + \sum_{v=1}^{V(\boldsymbol{w})} W_{h,v})} \prod_{v:W_{h,v}>0} \Gamma(W_{h,v}),$$

where $K(\boldsymbol{t}) = \sum_{k=1}^{\infty} \mathbb{I}_{\mathbb{N}_+}(T_k)$ and $H(\boldsymbol{s}) = \sum_{h=1}^{\infty} \mathbb{I}_{\mathbb{N}_+}(\sum_{k=1}^{\infty} S_{k,h})$ are the number of unique session-level and command-level topics respectively, and $V(\boldsymbol{w}) = \sum_{v=1}^{\infty} \mathbb{I}_{\mathbb{N}_+}(\sum_{h=0}^{\infty} W_{h,v})$ is the observed number of unique words.

4.1. *Bayesian inference with GEM priors.* Inference in the model with unbounded number of topics and vocabulary size can be carried out using a similar algorithm to the Gibbs sampling described in Section 3. Only minor modifications are required, since the marginals in (18) take a different form under the GEM priors. For example, for resampling the session-level topics, the probabilities in (10) are modified as follows:

$$p(t_d = r \mid \boldsymbol{t}^{-d}, \boldsymbol{w}, \boldsymbol{z}, \boldsymbol{s}) \propto \gamma^{\mathbb{I}_{\{K(\boldsymbol{t}^{-d})+1\}}(r)} (T_r^{-d})^{1 - \mathbb{I}_{\{K(\boldsymbol{t}^{-d})+1\}}(r)} \times \frac{\prod_{h:S_h^d>0} \tau^{\mathbb{I}_{\{0\}}(S_{r,h}^{-d})} (S_{r,h}^{-d})^{1 - \mathbb{I}_{\{0\}}(S_{r,h}^{-d})} \left[\prod_{\ell=2}^{S_h^d} (S_{r,h}^{-d} + \ell - 1)\right]^{\mathbb{I}_{\mathbb{N}_{>1}}(S_h^d)}}{\prod_{\ell=1}^{N_d} \{\tau + (\sum_{h=1}^{H(\boldsymbol{s})} S_{r,h}^{-d}) + \ell - 1\}},$$

where $r \in \{1, \ldots, K(\boldsymbol{t}^{-d}) + 1\}$, and the convention $0^0 = 1$ is adopted. Similarly, the probabilities in (11) for resampling command-level topics become:

$$p(s_{d,j} = \ell \mid \boldsymbol{s}^{-d,j}, \boldsymbol{w}, \boldsymbol{t}, \boldsymbol{z}) \propto \tau^{\mathbb{I}_{\{h:S_{t_d,h}=0\}}(\ell)} (S^{-d,j}_{t_d,\ell})^{1-\mathbb{I}_{\{h:S_{t_d,h}=0\}}(\ell)}$$
$$\times \frac{\prod_{v:W^{d,j}_v>0} \eta^{\mathbb{I}_{\{0\}}(W^{-d,j}_{\ell,v})} (W^{-d,j}_{\ell,v})^{1-\mathbb{I}_{\{0\}}(W^{-d,j}_{\ell,v})} \left[\prod_{q=2}^{W^{d,j}_v} (W^{-d,j}_{\ell,v} + q - 1)\right]^{\mathbb{I}_{\mathbb{N}_{>1}}(W^{d,j}_v)}}{\prod_{q=1}^{\sum_v W^{d,j}_v} \{\eta + (\sum_{v=1}^{V(\boldsymbol{w})} W^{-d,j}_{\ell,v}) + q - 1\}}$$
$$\times \frac{\prod_{q=1}^{Z^{d,j}} (\alpha_\ell + Z^{-d,j}_\ell + q - 1) \prod_{u=1}^{M_{d,j} - Z^{d,j}} (\alpha_0 + M^{*-d,j}_\ell - Z^{-d,j}_\ell + u - 1)}{\prod_{q=1}^{M_{d,j}} (\alpha_0 + \alpha_\ell + M^{*-d,j}_\ell + q - 1)},$$

where $\ell \in \{1, \ldots, H(\boldsymbol{s}^{-d,j}) + 1\}$. A modification is required also for the conditional probabilities of resampling the indicator $z_{d,j,i}$ in (15), resulting in:

$$p(z_{d,j,i} = b \mid \boldsymbol{z}^{-d,j,i}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{t}) \propto (\alpha_{s_{d,j}} + Z^{-d,j,i}_{s_{d,j}})^b (\alpha_0 + M^*_{s_{d,j}} - Z^{-d,j,i}_{s_{d,j}} - 1)^{1-b}$$
$$\times \left\{ \frac{\eta^{\mathbb{I}_{\{0\}}(W^{-d,j,i}_{0,w_{d,j,i}})} (W^{-d,j,i}_{0,w_{d,j,i}})^{1-\mathbb{I}_{\{0\}}(W^{-d,j,i}_{0,w_{d,j,i}})}}{\eta + \sum_{v=1}^{V(\boldsymbol{w}^{-d,j,i})} W^{-d,j,i}_{0,v}} \right\}^{1-b}$$
$$\times \left\{ \frac{\eta^{\mathbb{I}_{\{0\}}(W^{-d,j,i}_{s_{d,j},w_{d,j,i}})} (W^{-d,j,i}_{s_{d,j},w_{d,j,i}})^{1-\mathbb{I}_{\{0\}}(W^{-d,j,i}_{s_{d,j},w_{d,j,i}})}}{\eta + \sum_{v=1}^{V(\boldsymbol{w}^{-d,j,i})} W^{-d,j,i}_{s_{j,d},v}} \right\}^{b},$$

where $b \in \{0, 1\}$. The split-merge move in Section 3.4 can be equivalently extended to the model with GEM priors, using the same ideas presented in this section. Split-merge moves are expected to improve convergence of Gibbs sampling algorithms in models based on Dirichlet processes (Jain and Neal, 2004). Note that all the probabilities described in this section are extremely similar to the equations in Section 3, with the added complexity of handling previously unseen topics. Alternative MCMC algorithms for models based on the Dirichlet processes are extensively discussed in the literature (for example, Neal, 2000; Ishwaran and James, 2001; Teh et al., 2006).

5. Application to the Imperial College London honeypot data. The models described in Sections 2 and 4 are now applied to real data collected on a honeypot hosted within the Imperial College London (ICL) computer network. Within a time period between 21st May, 2021, and 27th January, 2022, the ICL honeypot collected approximately 40,000 unique sessions, observed over 1.3 million times. This is a large corpus of attacks for a single machine.

5.1. *Data preprocessing.* As discussed in the introduction, such sessions and commands must be *tokenised* to obtain the words and vocabulary. In this work, the tokenisation is performed with the *python* package NLTK (Bird, Klein and Loper, 2009), setting the regular expression `[a-zA-Z0-9_\.\-\*]+`. Also, commands observed in the ICL honeypot data often contain combinations of strings in hexadecimal form, preceded by the letter `x`. An example of such a command is

```
bin busybox echo -e x6b x61 x6d x69 dev dev .nippon
```

In these analyses, all such instances of hexadecimal strings (`x6b`, `x61`, `x6d` and `x69` in the above example) are replaced by the word `HEX`. Also, some commands display the word `GHILIMEA` appended to `HEX` strings. These are replaced with the word `GHILIMEA_word`. Similarly to standard preprocessing techniques in natural language processing, extremely

rare and extremely common words are removed from the dataset. For the ICL honeypot data, words appearing in less than 10 commands were removed, as were words appearing in over 10% of commands. Such words are often denoted *stopwords* in natural language processing and information retrieval (see, for example, Manning, Raghavan and Schütze, 2008). After preprocessing, a vocabulary of 1,003 unique words is obtained, and 2,617 uniquely observed sessions, for a total of 42,640 commands and 261,283 words. Each session has an average of 16.29 commands (with median 15), whereas each command contains on average 6.12 words (with median 2). Users who access the honeypot have malicious intent, and it is not uncommon to observe swear words and discriminatory language in many of the sessions. Those terms have been redacted in plots of the results.

5.2. *Topic estimation.* Before describing the results, some practical details about the estimation of topics from MCMC chains are discussed. In general, the number of session-level or command-level topics are unknown. The Dirichlet priors for $\boldsymbol{\lambda}$ and $\{\boldsymbol{\phi}_h\}$ in Section 2 assume fixed, pre-specified values of K and H. For inference with the Dirichlet prior, a maximum number of possible topics could be chosen, denoted $K_{\max}$ and $H_{\max}$, and the underlying number of topics could be estimated as the number of *non-empty* topics at each iteration of the MCMC sampling procedure. Furthermore, estimates of topic allocations based on the MCMC sampler described in Section 3 could be affected by the issue of label switching (Jasra, Holmes and Stephens, 2005). Therefore, session-level topic allocations are estimated in this work from the estimated posterior similarity between sessions i and j, calculated as $\sum_{s=1}^{M} \mathbb{1}_{t^\star_{i,s}}\{t^\star_{j,s}\}/M$, where M is the total number of posterior samples and $t^\star_{i,s}$ is the s-th sample for t_i. The posterior similarity matrix is invariant to permutations of the labels and therefore unaffected by label switching. After the posterior similarities are obtained for all pairs of sessions, hierarchical clustering with complete linkage is applied, with distance measure $1 - \hat{\pi}_{ij}$ (Medvedovic, Yeung and Bumgarner, 2004). A similar procedure could be followed for the command-level topics, but the very large number of commands would make the size of the similarity matrix unfeasible to calculate and store in memory on a machine. Therefore, the last sample from the MCMC chain is considered as the estimate of the command-level topics.

5.3. *Constrained topic modelling.* First, the constrained topic model in (1) is fitted on the postprocessed ICL honeypot data. Under the Dirichlet prior for $\boldsymbol{\lambda}$, the hyperparameter $\boldsymbol{\gamma}$ is set to $\boldsymbol{\gamma} = 0.1 \cdot \mathbf{1}_{K_{\max}}$, with $K_{\max} = 30$. The hyperparameter of the Dirichlet prior for the topic-specific word distribution is set to $\boldsymbol{\eta} = \mathbf{1}_{|V|}$. The MCMC sampler is run for 250,000 iterations with 50,000 burn-in, initialising the topics via spectral clustering with $K_{\max}$ clusters. The results are displayed in Figure 2. The session-level topics are estimated using the procedure described in Section 5.2. Figure 2a plots the resulting barplot of topic frequencies of estimated session-level topics, for a number of topics equal to the modal number of non-empty topics, $\hat{K}_{\varnothing} = 20$. Furthermore, Figure 2b displays the barplot of the estimated distribution for the number of non-empty topics. The barplot is also compared to the distribution obtained under a GEM prior for $\boldsymbol{\lambda}$, with hyperparameter $\gamma = 3$, corresponding to $K_{\max} \times 0.1$, using the same setup for the MCMC sampler. The resulting distributions show agreement, demonstrating a similar performance of the Dirichlet and GEM priors in estimating the modal number of topics. In general, the interplay between the prior parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ appears to have an effect on the number of *small* clusters that are estimated from the data: if $\boldsymbol{\eta}$ increases, the clusters in the right tail of Figure 2a tend to be incorporated within the larger clusters. On the other hand, if $\boldsymbol{\eta}$ decreases towards zero, it is expected to estimate more topics.

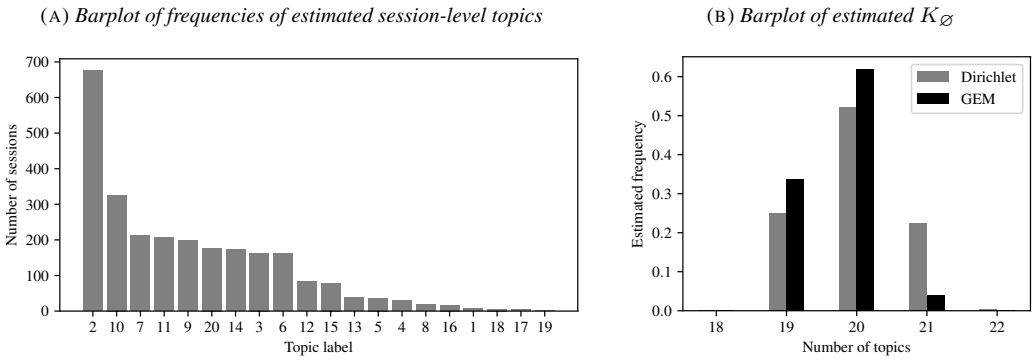


FIG 2: *Estimated topic frequencies and estimated distribution the number of non-empty topics $K_\varnothing$ under the constrained topic model in (1), fitted on the ICL honeypot data.*

5.3.1. *Results: the model discovers a rare and unusual MIRAI variant.* The inferred meaning of each topic is summarised in Table 1, according to the type of sessions that are assigned to each group. The clusters appear to mostly contain botnets and different *variants* of MIRAI malware. MIRAI is a type of botnet first emerged in 2016, which was specifically targeted towards compromising Internet of Things (IoT) devices, or launching Distributed Denial of Service (DDoS) attacks. Recently, it has been repurposed for Bitcoin mining on IoT devices compromised via brute-force attacks on protocols such as SSH and Telnet. Over the years, many different variants of MIRAI have emerged, and other bots with similar structure (Lingenfelter, Vakilinia and Sengupta, 2020; Sadique and Sengupta, 2021; Zhu et al., 2022), which appear to be assigned to different topics in Table 1.

Interestingly, careful examination of the sessions assigned to *topic 5* estimated via the constrained topic model in (1) helped analysts to discover an rare and unusual variant of MIRAI, called MinerFinder (Bevington, 2021). The objective of MinerFinder is to look for existing coin miner configurations, and try to gain root privileges to take control of the miner infrastructure, if found. This demonstrates that the constrained topic model with a single topic per session, even in its simplest form, could be extremely helpful for analysts to discover new attack patterns. Within *topic 5*, MinerFinder is also mixed with other more common MIRAI variants, which share a common frequency distribution of words. Ideally, a clustering algorithm should be able to single-out MinerFinder from other MIRAI variants, despite their similarities. This might be possible when a hierarchical structure is added to the topics, and the command structure is explicitly used, as demonstrated in Section 5.5.

Overall, most of the activity on the honeypot seems to be related to attempts to install botnets or coin miners, but topics show remarkable separation between sessions and corresponding intents, as demonstrated in the list of malware and objectives in Table 1. Further intuition about the differences between topics could be provided by examining the topic-specific word distributions, which can be displayed via wordclouds, plotted in Figure 3. The figure shows that words within each topic are fairly heterogeneous, but a number of words appear frequently across multiple topics, such as `HEX`, `cd` or `sh`. A potential solution to better visualise and further differentiate topic-specific word distributions would be to introduce a *secondary topic*, which would capture the distribution of the most common words shared across multiple topics and sessions. This solution is explored in the next section.

5.4. *Constrained topic modelling with a secondary topic.* As discussed in the previous section, a possible solution to aid topic interpretability and further discriminate topic-specific word distributions would be to add a secondary topic to the model. In this section, the constrained topic model with secondary topic in (2) is therefore fitted to the ICL honeypot data.

TABLE 1

Estimated session-level topics and corresponding intent under the constrained topic model in (1).

Topic label	Type of malware	Objective
1	Shellbot	Install bot
2	(ptmx) unnamed botnet, MIRAI	Gather system information, change permissions, execute MIRAI variants
3	MIRAI	Download and execute MIRAI variants kura and kurc, fingerprint system
4	MIRAI	Download sora malware, write upnp and updDl malware via echoing HEX strings
5	MIRAI, **MinerFinder (new variant)**	Download malware, change permissions, gather system information, fingerprint system
6	Shellbot, SBIDIOT, coin miner	Download and execute coin miner and MIRAI malware, change SSH keys
7	(s4y, LAYER) unnamed botnet, MIRAI	Gather system information, change permissions, execute MIRAI variants
8	Coin miner	Download and execute coin mining malware
9	MIRAI	Download and execute MIRAI variants PEDO, ECCHI, PEACH...
10	MIRAI	Download and execute MIRAI variants tftp1.sh, tftp2.sh...
11	MIRAI	Determine shell executable, check busybox is present, print error message to console
12	(misa) unnamed botnets	Gather system information, change permissions, execute MIRAI variants
13	Shellbot, coin miner	Scan system, look for GPUs, look for coin miners, download malware
14	MIRAI	Download and execute MIRAI variants sora, Pemex...
15	MIRAI	Download and execute MIRAI variant DNXFCOW via echoing single HEX strings
16	MIRAI	Download and execute MIRAI variant DNXFCOW via echoing multiple HEX strings
17	MikroTik bot, coin miner	Gather system information, gather MikroTik router information, look for coin miners
18	GHILIMEA, PentaMiner coin miner script	Install coin miner, kill mining processes with high CPU usage in order to go undetected
19	MikroTik bot	Attempt to gain access to MikroTik router
20	Hive OS attack, coin miner	Download miner, attempt to take over configurations in Hive OS mining platform

FIG 3: *Wordclouds of estimated topic-specific word distributions under the constrained topic model in (1), fitted on the ICL honeypot data.*

The setup of the MCMC sampler is chosen to be identical to the previous section, and the additional hyperparameters are set to $\alpha_0 = 0.1$ and $\alpha_k = 0.9$ for $k = 1, \ldots, K_{\max}$, resulting in a prior probability of 90% for a word to be allocated to a primary topic. The sampler is ini-

FIG 4: *Wordclouds of estimated topic-specific word distributions under the constrained topic model with secondary topic in* (2)*, fitted on the ICL honeypot data.*

tialised using the topics estimated by the constrained topic model in (1), fitted in Section 5.3. In order to suitably compare the topic-specific word distributions and avoid the label switching issue in clustering (Jasra, Holmes and Stephens, 2005), the MCMC chain for model (2) was initialised using the output from model (1) obtained in the previous section. The indicators $z_{d,j,i}$ are initialised from a Bernoulli distribution with probability equal to the proportion of documents in which the word $w_{d,j,i}$ occurred. The resulting wordclouds for some of the topic-specific word distributions are plotted in Figure 4, including the distribution of the secondary topic, labelled *topic 0*. Note that with a secondary topic, words are implicitly sampled from the mixture distribution $\tilde{\phi}_{t_d} = \theta_{t_d}\phi_{t_d} + (1-\theta_{t_d})\phi_0$. This aids interpretability of the latent intent of the session-level topic t_d, since the common words shared across most topics are filtered out and included in ϕ_0 instead. This is confirmed when comparing Figure 4 with Figure 3 (obtained from a model that *does not include* a secondary topic). Overall, Figure 4 shows that the secondary topic captures the most common words across documents, such as the string `HEX` and simple shell commands, for example `cd`, `chmod`, `var`, `echo`, `shell` and `tmp`. When comparing with Figure 3, the word distributions in Figure 4 appear to be less dominated by common words. For example, for topic 1, the words `tmp` and `cd` appear to be among the most representative words in Figure 3, but they have a much less prominent role in the wordcloud representation in Figure 4, where words such as `x86_64`, `uname` and `Xorg` gain importance. These words appear to be much more representative of the actual intents of the session, which is particularly helpful when communicating results to analysts.

Overall, the assumption of having only one topic per session might be limiting, even if an additional secondary shared topic is added. This is mainly because most sessions could be considered as mixtures of commands, where each command has its own intent. Therefore, a more precise clustering of topics and commands might be provided by the Hierarchical Constrained Topic Model (HTCM) in Section 2.2, which is considered in the next section.

5.5. *Hierarchical Constrained Topic Modelling (HCTM).* As discussed in the previous section, the HCTM in Section 2.2 could help to further elucidate the underlying group structure within the ICL honeypot data. Similarly to Section 5.3, the MCMC is run for 250,000 iterations with 50,000 burn-in. The command-level and session-level topics are initialised via the spectral clustering algorithm described in Section 3.5, setting Dirichlet priors of dimension $K_{\max} = 50$ and $H_{\max} = 50$, with hyperparameters $\boldsymbol{\eta} = \mathbf{1}_{|V|}$, $\boldsymbol{\tau} = 0.1 \cdot \mathbf{1}_{H_{\max}}, \boldsymbol{\gamma} = 0.1 \cdot \mathbf{1}_{K_{\max}}$. The session-level and command-level topics are estimated following the procedure described in Section 5.2, setting $\hat{K}_{\varnothing} = 36$ and $\hat{H}_{\varnothing} = 38$, corresponding to the modal number of non-empty topics. Figure 5 displays the frequency distribution of the estimated session-level (Figure 5a) and command-level topics (Figure 5c), followed by the estimated distributions of the number of non-empty session-level (Figure 5b) and command-level topics (Figure 5d).

The wordclouds for the resulting session-level and command-level topic-specific distributions are displayed in Figure 6. Figure 6a displays the topic-specific word distributions

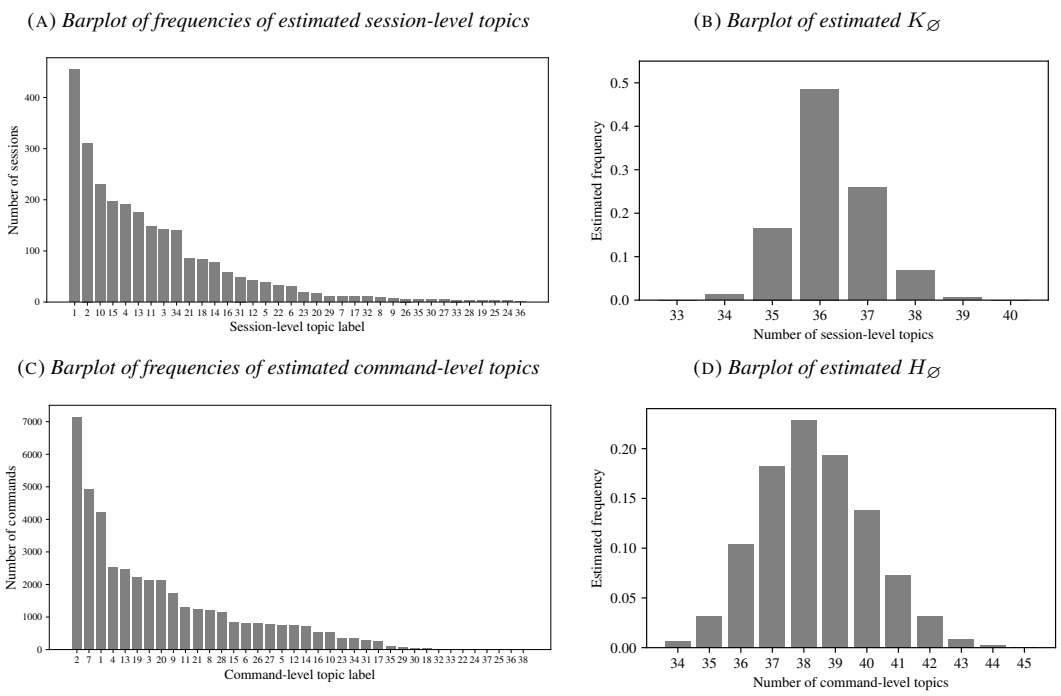


FIG 5: *Frequency distributions of the estimated session-level and command-level topics, and estimated distribution the number of non-empty session-level topics $K_\varnothing$ and command-level topics $H_\varnothing$, under the hierarchical constrained topic model in (3), fitted on the ICL honeypot data.*

obtained from the estimated session-level topics, whereas Figure 6b plots the wordclouds corresponding to the estimated command-level topics. Interestingly, when compared to Figure 3, the word distributions in Figure 6 appear more heterogeneous, especially at the command-level. This is not surprising: the constrained topic model in (1) gives the *same* primary topic to all words in a session, whereas the HCTM admits command-specific topics. In the ICL honeypot data, and more generally in session data, individual commands tend to have a specific intent identified by specific words in the command (for example, `wget` for downloading files from a web server under the HTTP, HTTPS, and FTP protocols). Therefore, having command-specific topics helps in identifying the intents of individual commands, making the command-level topic-specific word distributions highly interpretable.

Similarly to the previous section, the intents for the estimated session-level and command-level topics are summarised in Table 2 and 3. In general, the two tables show that some topics correspond to the same intent, achieved through different words or commands. In particular, Table 2 shows the same malware types as Table 1, with similar objectives. Similarly to the results in the previous section, different MIRAI variants are observed, and malware types such as shellbots, coin miners, the Hive OS attack and the MikroTik bot are all allocated to separate topics. In addition to the session-level objectives in Table 1, Table 2 also shows topics representing reconnaissance sessions, where intruders attempted to gather system information, for example by checking directories and determining shell executables. MinerFinder is again discovered and relevant sessions are singled-out in the topic with label 24. In particular, under the hierarchical constrained topic model, MinerFinder is explicitly split from the similar MIRAI variants that were present in *topic 5* under the constrained topic model (*cf.* Table 1), which are instead allocated to *topic 22* under the HCTM (*cf.* Table 2). It must be remarked that MinerFinder is *not detected* using alternative clustering approaches based on spectral clustering or standard LDA fitted via *gensim* (*cf.* Section 3.5). If the topics are

(A) *Session-level topic-specific word distributions*

(B) *Command-level topic-specific word distributions*

FIG 6: *Wordclouds of estimated session-level and command-level topic-specific word distributions under the hierarchical constrained topic model in (3), fitted on the ICL honeypot data.*

TABLE 2
Estimated session-level topics and corresponding intent under the hierarchical constrained topic model in (3).

Topic label	Type of malware	Objective
1	MIRAI	Check shell and directories, download and execute MIRAI malware, delete files
2	(ptmx) unnamed botnet, MIRAI	Gather system information, change permissions, execute MIRAI variants
3	Shellbot, coin miner, MIRAI	Download and execute coin miner and MIRAI malware
4	MIRAI	Determine shell executable, check busybox is present, print error message to console
5	Shellbot, coin miner	SSH backdoor botnet, download malware, change SSH keys, check CPU/GPU information
6	MIRAI	Download and execute MIRAI malware (for example, garm or gmips), delete files
7	Shellbot, coin miner	Gather CPU/GPU information, download coin miners, kill existing coin miners
8	Reconnaissance	Check mounted file system, gather system information, fingerprint system
9	MIRAI	Write malware (for example dvrHelpwer and updDl) via echoing HEX strings
10	Reconnaissance	Check shell and directories, delete files (.ptmx, Switchblades)
11	Hive OS attack, coin miner	Download miner, attempt to take over configurations in Hive OS mining platform
12	MIRAI	Execute MIRAI variant PEDO, fingerprint system
13	MIRAI	Execute MIRAI variants kura and kurc, fingerprint system
14	MIRAI	Determine shell, download and execute MIRAI variant DNXFCOW via echoing HEX strings
15	Reconnaissance	Check shell and directories, delete files (.ptmx, .s4y)
16	MIRAI	Download and execute MIRAI variant PEDO and mika, fingerprint system
17	Coin miner	Download coin miner (c3pool)
18	MIRAI	Download, execute MIRAI variant ECCHI, delete files, fingerprint system
19	MIRAI	Download malware, write updDl malware via echoing HEX strings
20	MIRAI	Download sora malware, write upnp and updDl malware via echoing HEX strings
21	MIRAI	Gather system information, change permissions, execute MIRAI variants
22	MIRAI	Download malware, change permissions, gather system information, fingerprint system
23	Coin miner, SBIDIOT	Download and execute coin mining malware
24	**MinerFinder (new MIRAI variant)**	Changes SSH keys, looks for coin miners, attempt to take over miners
25	MIRAI	Check shell and directories, execute MIRAI malware (Skyline and Akim), delete files
26	GHILIMEA, PentaMiner coin miner script	Install coin miner, kill mining processes with high CPU usage in order to go undetected
27	Reconnaissance	Attempt to read and change SSH keys
28	MIRAI	Write RONALD malware via echoing HEX strings
29	MIRAI	Gather system information, print error message (component of DNXFCOW MIRAI variant)
30	MIRAI	Gather system information, change permissions, execute MIRAI variant cowffxxna
31	MikroTik bot, coin miner	Gather system information, gather MikroTik router information, kill existing coin miners
32	MIRAI	Download and execute MIRAI variant DNXFCOW via echoing multiple HEX strings
33	Shellbot, coin miner	Scan system, look for GPUs, look for coin miners, download malware
34	MIRAI	Gather system information, change permissions, execute MIRAI variants
35	Coin miner	Download coin miner (TeamTNT)
36	MikroTik bot	Remove firewall NAT rules on MikroTik router

estimated using the procedures described in Section 3.5, MinerFinder is usually allocated to large clusters, with a number of sessions ranging between 80 and 1,000.

Furthermore, Figure 7 displays the heatmap of Jaccard similarity scores comparing the results of the constrained model (*cf.* Section 5.3) and the hierarchical constrained topic model fitted in this section, demonstrating significant agreement. The level of agreement is remarkable, considering that the session-level topics are obtained under two different modelling assumptions: in Section 5.3, the constrained topic model in (1) directly uses the session-level topic to obtain the word distribution for the entire session. On the other hand, the session-level topics in HTCMs are only estimated from sequences of command-level topics, which are themselves unknown and therefore estimated.

More details about the commands appearing in the sessions are given in Table 3, and such objectives can be associated with the command-level topic-specific wordclouds in Figure 6b. Overall, analysing the results of the HTCM confirms the results obtained in the previous section, with the added benefit of having estimated command-level topics.

6. Conclusion. Models for clustering session data collected on honeypots were proposed, with the objective of finding groups of similar attacks which could aid cyber analysts in detecting emerging intrusion attempts and vulnerabilities. The proposed models are based on modifications of Latent Dirichlet Allocation, aimed at improving topic interpretability and convergence properties. In particular, the concepts of primary and secondary topics were introduced, along with session-level and command-level topics. Secondary topics are used to represent common, high-frequency words, whereas the primary topic is used for the words which define the latent intent of the session. Furthermore, two hierarchical layers of topics were introduced: a command-level topic which determines the word distribution, and

TABLE 3

Estimated command-level topics and corresponding intent under the hierarchical constrained topic model in (3).

Topic label	Objective
1	Attempt to write file `.ptmx` or `LAYER` to directory `var`, `run` or `tmp` and change directory if writeable
2	Attempt to write file `.ptmx` to directory `netslink`, `mnt` or `shm` and change directory if writeable
3	Attempt to copy, write and delete files
4	Attempt to grant full permissions to all users on malware files
5	Gather system information about CPU architectures
6	Kill processes, gather system information, Hive OS logon attempt
7	Check available commands, determine shell executable
8	Check if `busybox` exists, print error message to console (component of `PEDO` MIRAI variant)
9	Gather system information about mounted file systems, copy files
10	Fingerprint readable and writeable directories to hidden file `.nippon`
11	Download malware from internet using `wget` and `curl`
12	Attempt to read and delete files
13	Attempt to execute malware, delete files after execution
14	Check if directories such as `var`, `run` or `tmp` exist, by attempting to change directory
15	Download malware using `tftp`
16	Download, execute and delete malware files
17	Download malware using `ftpget`
18	Download and install GHILIMEA coin mining malware, kill own processes if CPU usage is high
19	Check if `busybox` exists, check available commands, determine shell executable
20	Check if `busybox` exists, print error message to console (component of `KURA` and `OWARI` MIRAI variants)
21	Check if `busybox` exists, print error message to console (component of `ECCHI` MIRAI variant)
22	Download malware to attempt to compromise MikroTik router
23	Look for existing coin miners, gather GPU and CPU information
24	Attempt to gather information about MikroTik router
25	Determine shell executable
26	Attempt to write file `.file` to directory `netslink`, `mnt` or `boot` and change directory if writeable
27	Copy, grant full permission, execute and delete `.cowbot` malware (MIRAI variant)
28	Write malware binary to disk via echoing `HEX` bits
29	Download coin miner malware from `c3pool`
30	Read SSH authorised keys, attempt to delete and replace authorised keys
31	Exit shell (part of GHILIMEA coin mining malware)
32	Install GHILIMEA coin mining malware, kill own processes if CPU usage is high
33	Check internet connectivity, check if GHILIMEA is already installed, kill own processes if CPU usage is high
34	Read and delete files (for example, `.none` and `.human`)
35	Check if `busybox` exists, print error message to console (component of `RONALD` and `Akim` MIRAI variant)
36	Remove firewall NAT rules on MikroTik router
37	Remove temporary directory `p` (singleton cluster)
38	Exit shell (singleton cluster)

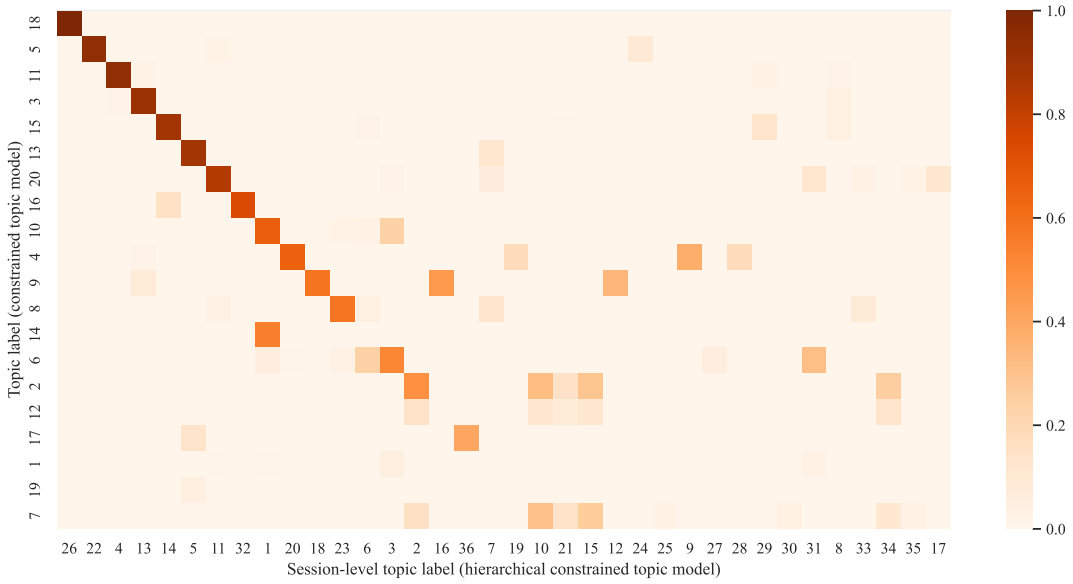


FIG 7: *Heatmap of Jaccard similarities between the sessions assigned to session-level topics under the constrained topic model (cf. Section 5.3) and hierarchical constrained topic modelling (cf. Section 5.5).*

a session-level topic which controls the distribution of the command-level intents. Furthermore, the proposed methodologies extend to a Bayesian nonparametric framework, admitting an unbounded (and unknown) number of latent intents.

The proposed models could be further extended by introducing dependencies between command-level topics. In particular, a Markov model with H states could be devised, whereby each session-level topic corresponds to different transition probabilities between command-level topics. Also, a possible extension of this work could consider dynamically-evolving topics (dynamic topic modelling, Blei and Lafferty, 2006), or explicitly encode correlation between topics (correlated topic models, Lafferty and Blei, 2005), or a combination of the two approaches (see, for example, Tomasi et al., 2020).

Sections 1.1 and 5.1 briefly discussed some of the challenges related to tokenisation in cyber-security applications. Depending on the chosen tokenisation method, important context about the commands might be lost, which might prevent cyber-security analysts to make rapid decisions and assessments on the content of the session. One possibility in this direction would be to explore deep neural network methods, inspired by their use for large language models. For example, the RAPIDS CLX library *cyBERT*[2] uses deep neural networks and transformers for parsing cyber-security logs, including session data.

Overall, session data collected on honeypots are a valuable resource for cyber analysts, and principled statistical modelling is required to obtain actionable insights from these data. The proposed methodologies have provided useful groupings of the attacks observed on a university network, including the discovery of an unusual MIRAI variant which attempts to take over existing coin miner infrastructure. This demonstrates the potential of topic models to elucidate hidden structure within text data, obtaining valuable insights for cyber-defence purposes.

Acknowledgements. The authors thank Andy Thomas and the Information & Communications Technology team at Imperial College London for their support on data processing. FSP and NAH acknowledge funding from Microsoft Security, through the research grant *"Understanding the enterprise: Host-based event prediction for automatic defence in cyber-security"*. AM and NAH acknowledge funding from the Data-centric Engineering programme at the Alan Turing Institute, for the Grand Challenge *"Monitoring Complex Systems"*. DG acknowledges funding from the Department of Mathematics at Imperial College London. Part of this work was carried out when AM was a postdoctoral research associate in statistical cyber-security at Imperial College London and the Alan Turing Institute. PT is now a Quantitative Analyst at Abios, and he completed part of this work as part of a masters' degree at Imperial College London.

Code. A *python* library that implements the methodologies proposed in this article is available in the Github repository `fraspass/lda_clust`.

REFERENCES

AGGARWAL, C. C. and ZHAI, C. (2012). A Survey of Text Classification Algorithms. In *Mining Text Data* 163–222. Springer US.

ALLISON, B., GUTHRIE, D. and GUTHRIE, L. (2006). Another Look at the Data Sparsity Problem. In *Proceedings of the 9th International Conference on Text, Speech and Dialogue*. *TSD'06* 327–334. Springer-Verlag, Berlin, Heidelberg.

ARCHAMBEAU, C., LAKSHMINARAYANAN, B. and BOUCHARD, G. (2015). Latent IBP Compound Dirichlet Allocation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **37** 321-333.

[2] see `https://github.com/rapidsai/clx`.

ASCOLANI, F., LIJOI, A., REBAUDO, G. and ZANELLA, G. (2022). Clustering consistency with Dirichlet process mixtures. *Biometrika (to appear)*.

BALIKAS, G., AMINI, M.-R. and CLAUSEL, M. (2016). On a Topic Model for Sentences. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. *SIGIR'16* 921–924. Association for Computing Machinery, New York, NY, USA.

BEVINGTON, R. (2021). Microsoft Sentinel Blog: Unusual MIRAI variant looks for mining infrastructure. URL: https://techcommunity.microsoft.com/t5/microsoft-sentinel-blog/unusual-mirai-variant-looks-for-mining-infrastructure/ba-p/2756669.

BIRD, S., KLEIN, E. and LOPER, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

BLEI, D. M. and LAFFERTY, J. D. (2006). Dynamic Topic Models. In *Proceedings of the 23rd International Conference on Machine Learning*. *ICML '06* 113–120. Association for Computing Machinery, New York, NY, USA.

BLEI, D. M., NG, A. Y. and JORDAN, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research* **3** 993–1022.

CAI, D., CAMPBELL, T. and BRODERICK, T. (2021). Finite mixture models do not reliably learn the number of components. In *Proceedings of the 38th International Conference on Machine Learning*. *Proceedings of Machine Learning Research* **139** 1158–1169. PMLR.

CAI, D., HE, X. and HAN, J. (2011). Locally Consistent Concept Factorization for Document Clustering. *IEEE Transactions on Knowledge and Data Engineering* **23** 902-913.

CRESPI, V., HARDAKER, W., ABU-EL-HAIJA, S. and GALSTYAN, A. (2021). Identifying botnet IP address clusters using natural language processing techniques on honeypot command logs. *arXiv e-prints*.

DAHL, D. B. (2003). An improved merge-split sampler for conjugate Dirichlet process mixture models Technical Report No. 1086, Department of Statistics, University of Wisconsin, Madison.

DESHMUKH, S., RADE, R. and KAZI, D. F. (2019). Attacker behaviour profiling using stochastic ensemble of Hidden Markov models. *arXiv e-prints*.

DING, R., NALLAPATI, R. and XIANG, B. (2018). Coherence-Aware Neural Topic Modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* 830–836. Association for Computational Linguistics, Brussels, Belgium.

DOSHI-VELEZ, F., WALLACE, B. C. and ADAMS, R. (2015). Graph-Sparse LDA: A Topic Model with Structured Sparsity. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. *AAAI'15* 2575–2581. AAAI Press.

HIGHNAM, K., ARULKUMARAN, K., HANIF, Z. and JENNINGS, N. R. (2021). BETH Dataset: Real Cybersecurity Data for Anomaly Detection Research. *ICML Workshop on Uncertainty and Robustness in Deep Learning*.

HUBERT, L. and ARABIE, P. (1985). Comparing partitions. *Journal of Classification* **2** 193–218.

ISHWARAN, H. and JAMES, L. F. (2001). Gibbs Sampling Methods for Stick-Breaking Priors. *Journal of the American Statistical Association* **96** 161-173.

JAIN, S. and NEAL, R. M. (2004). A Split-Merge Markov chain Monte Carlo Procedure for the Dirichlet Process Mixture Model. *Journal of Computational and Graphical Statistics* **13** 158-182.

JASRA, A., HOLMES, C. C. and STEPHENS, D. A. (2005). Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture Modeling. *Statistical Science* **20** 50 – 67.

JIANG, H., ZHOU, R., ZHANG, L., WANG, H. and ZHANG, Y. (2019). Sentence level topic models for associated topics extraction. *World Wide Web* **22** 2545–2560.

KE, Z. T. and WANG, M. (2022). Using SVD for Topic Modeling. *Journal of the American Statistical Association (to appear)*.

LAFFERTY, J. and BLEI, D. (2005). Correlated Topic Models. In *Advances in Neural Information Processing Systems* (Y. WEISS, B. SCHÖLKOPF and J. PLATT, eds.) **18**. MIT Press.

LINGENFELTER, B., VAKILINIA, I. and SENGUPTA, S. (2020). Analyzing Variation Among IoT Botnets Using Medium Interaction Honeypots. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)* 0761-0767.

LIU, J. S. (1994). The Collapsed Gibbs Sampler in Bayesian Computations with Applications to a Gene Regulation Problem. *Journal of the American Statistical Association* **89** 958-966.

MANNING, C. D., RAGHAVAN, P. and SCHÜTZE, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

MEDVEDOVIC, M., YEUNG, K. Y. and BUMGARNER, R. E. (2004). Bayesian mixture model based clustering of replicated microarray data. *Bioinformatics* **20** 1222–1232.

MILLER, J. W. and HARRISON, M. T. (2013). A simple example of Dirichlet process mixture inconsistency for the number of components. In *Advances in Neural Information Processing Systems* (C. J. C. BURGES, L. BOTTOU, M. WELLING, Z. GHAHRAMANI and K. Q. WEINBERGER, eds.) **26**. Curran Associates, Inc.

MILLER, J. W. and HARRISON, M. T. (2014). Inconsistency of Pitman-Yor Process Mixtures for the Number of Components. *Journal of Machine Learning Research* **15** 3333-3370.

NEAL, R. M. (2000). Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of Computational and Graphical Statistics* **9** 249-265.

PITMAN, J. (2006). *Combinatorial Stochastic Processes*, 1 ed. *Ecole d'Eté de Probabilités de Saint-Flour XXXII*. Springer-Verlag Berlin Heidelberg.

RADE, R., DESHMUKH, S., NENE, R., WADEKAR, A. S. and UNNY, A. (2018). Temporal and stochastic modelling of attacker behaviour. In *International Conference on Intelligent Information Technologies* 30–45. Springer.

SADIQUE, F. and SENGUPTA, S. (2021). Analysis of Attacker Behavior in Compromised Hosts During Command and Control. In *ICC 2021 - IEEE International Conference on Communications* 1-7.

SANNA PASSINO, F. and HEARD, N. A. (2020). Bayesian estimation of the latent dimension and communities in stochastic blockmodels. *Statistics and Computing* **30** 1291–1307.

SATO, I. and NAKAGAWA, H. (2010). Topic Models with Power-Law Using Pitman-Yor Process. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. *KDD'10* 673–682. Association for Computing Machinery, New York, NY, USA.

SETHURAMAN, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica* **4** 639–650.

SHRIVASTAVA, R. K., BASHIR, B. and HOTA, C. (2019). Attack detection and forensics using honeypot in IoT environment. In *International Conference on Distributed Computing and Internet Technology* 402–409. Springer.

TEH, Y. W., JORDAN, M. I., BEAL, M. J. and BLEI, D. M. (2006). Hierarchical Dirichlet Processes. *Journal of the American Statistical Association* **101** 1566-1581.

TOMASI, F., CHANDAR, P., LEVY-FIX, G., LALMAS-ROELLEKE, M. and DAI, Z. (2020). Stochastic Variational Inference for Dynamic Correlated Topic Models. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*. *Proceedings of Machine Learning Research* **124** 859–868. PMLR.

ŘEHŮŘEK, R. and SOJKA, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* 45-50.

WALLACH, H. M. (2008). Structured Topic Models for Language, PhD thesis, University of Cambridge.

WILLIAMSON, S., WANG, C., HELLER, K. A. and BLEI, D. M. (2010). The IBP Compound Dirichlet Process and Its Application to Focused Topic Modeling. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*. *ICML'10* 1151–1158. Omnipress, Madison, WI, USA.

WILSON, A. T. and CHEW, P. A. (2010). Term Weighting Schemes for Latent Dirichlet Allocation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. *HLT'10* 465–473. Association for Computational Linguistics, USA.

XIE, P. and XING, E. P. (2013). Integrating Document Clustering and Topic Modeling. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. *UAI'13* 694–703. AUAI Press, Arlington, Virginia, USA.

XU, W., LIU, X. and GONG, Y. (2003). Document Clustering Based on Non-Negative Matrix Factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. *SIGIR'03* 267–273. Association for Computing Machinery, New York, NY, USA.

ZHAI, K. and BOYD-GRABER, J. (2013). Online Latent Dirichlet Allocation with Infinite Vocabulary. In *Proceedings of the 30th International Conference on Machine Learning* (S. DASGUPTA and D. MCALLESTER, eds.). *Proceedings of Machine Learning Research* **28** 561–569. PMLR, Atlanta, Georgia, USA.

ZHANG, X. (2020). Bayesian Latent Feature Modelling for Unstructured Data, PhD thesis, Imperial College London.

ZHU, Y., CHEN, Z., YAN, Q., WANG, S., LI, E., PENG, L. and ZHAO, C. (2022). Mining Function Homology of Bot Loaders from Honeypot Logs. *arXiv e-prints*.

APPENDIX A: SIMULATIONS AND RESULTS ON SYNTHETIC DATA

In this section, the proposed models are compared and contrasted on synthetic datasets, in order to assess their performance in recovering the session-level topics, the main quantity of inferential interest. Note that the true topics are *known* when data are simulated. If synthetic data are used, the true underlying session-level topics are available, and the true allocations could be compared to the estimated topics via the Adjusted Rand Index (ARI, Hubert and Arabie, 1985). Each simulation is repeated for 50 datasets using different seeds, setting $|V| = 50,\ D = 100,\ N_d = 10,\ M_{d,j} = 10,\ \boldsymbol{\lambda} = 1/K \cdot \mathbf{1}_K$ for each simulated dataset. The MCMC sampler is run for 25,000 iterations with 10,000 burn-in, initialising the topics via spectral clustering. Unless otherwise specified, the hyperparameter $\boldsymbol{\gamma}$ is set to $\boldsymbol{\gamma} = 0.1 \cdot \mathbf{1}_{K_{\max}}$.

(A) *Boxplot of ARIs for estimated session-level topics* (B) *Barplot of estimated number of non-empty topics* $K_{\varnothing}$

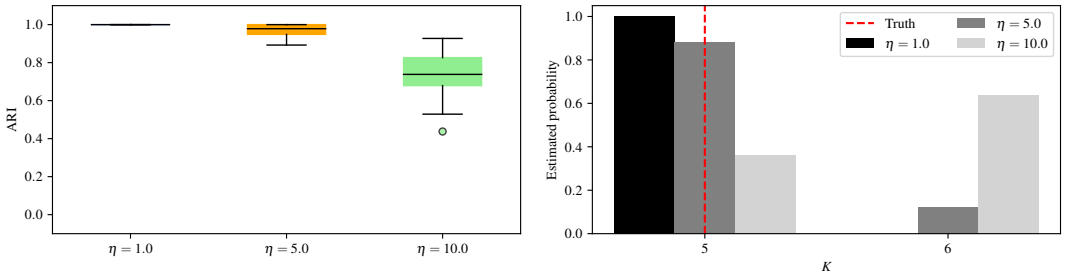


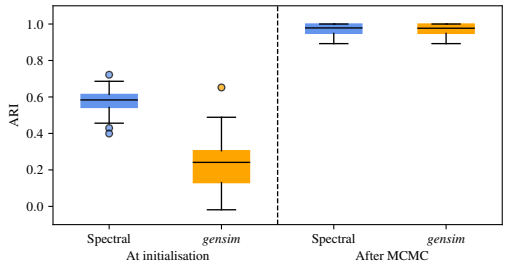
FIG 8: *Summary plots obtained from 50 simulated datasets from model (1), with* $|V| = 50$, $K = 5$, $D = 100$, $N_d = 10$, $M_{d,j} = 10$, $\boldsymbol{\lambda} = 1/K \cdot \mathbf{1}_K$, *using different values for* $\boldsymbol{\eta}$, *such that* $\boldsymbol{\eta} = \eta \cdot \mathbf{1}_K$. *The MCMC sampler is run for 25,000 iterations with 10,000 burn-in, setting* $K_{max} = 10$.

First, datasets are simulated from model (1), setting $K = 5$ and using three different values of the parameter $\boldsymbol{\eta}$. If $\boldsymbol{\eta} = \eta \cdot \mathbf{1}_K$, the parameter η controls the *spikiness* of the topic-specific word distributions: low values of η correspond to distributions which assign most of the probability mass to a small number of words, whereas larger values of η correspond to distributions where the probability mass is distributed more uniformly across words. In the MCMC sampler, K_{max} is set to 10, implying that the sampler should identify 5 empty topics. The results are reported in Figure 8: Figure 8a reports the ARIs for the estimated session-level topics, whereas Figure 8b shows the barplot of the estimated number of non-empty topics, denoted $K_{\varnothing}$. As expected, the ARI decreases when the value of η increases, since the topic-specific distributions become increasingly uniform and therefore similar. Increasing the value of η also provides worse estimates of the true number of topics used in the simulation. Ideally, $K_{\varnothing}$ should correspond to the value of K used in the simulation, provided that K_{max} in the MCMC algorithm is chosen to be larger than the true underlying K.

Next, inference is repeated on the 50 datasets simulated as in Figure 8 with $\boldsymbol{\eta} = 5 \cdot \mathbf{1}_K$, using different initialisation methods and priors for $\boldsymbol{\lambda}$. In particular, results are compared between the spectral and *gensim* initialisation schemes described in Section 3.5. Figure 9a displays the boxplots of ARIs for the session-level topics across the different datasets, after initialisation, before and after running the MCMC sampling scheme. The plot shows that the spectral initialisation scheme appears to have a better performance initially, but both methods lead to equivalent results after MCMC sampling. Furthermore, results on estimation of the number of topics are compared between two different priors: a K_{max}-dimensional Dirichlet distribution with $K_{\text{max}} = 10$ and $\boldsymbol{\gamma} = 0.1 \cdot \mathbf{1}_{K_{\text{max}}}$ (also used in Figure 8b) or a GEM prior on $\boldsymbol{\lambda}$ with $\gamma = 0.1$. Figure 9b reports the resulting barplot for the estimated modal number of non-empty communities across the different datasets, suggesting that the two priors have a similar performance.

Second, datasets are simulated from model (2), using $\boldsymbol{\eta} = \mathbf{1}_K$, $K = 5$ and different values of θ_k. The value of θ_k corresponds to the expected proportion of words sampled from the primary topic. Therefore, small values of θ_k are expected to make inferential procedures more difficult, since less observations are available from the primary topics, and words are sampled instead from a secondary topic, shared across sessions and commands. Furthermore, the secondary topic makes the primary topics more difficult to estimate, since the words are implicitly sampled from the mixture distribution $\tilde{\boldsymbol{\phi}}_{t_d} = \theta_{t_d}\boldsymbol{\phi}_{t_d} + (1 - \theta_{t_d})\boldsymbol{\phi}_0$. If θ_k decreases for all k, the distributions $\tilde{\boldsymbol{\phi}}_1, \ldots, \tilde{\boldsymbol{\phi}}_K$ become increasingly similar, and a drop in the ARI similar to Figure 8a is expected. In the MCMC sampler, the required parameters are set to $K_{\text{max}} = 10$, $\alpha_h = 0.9$ and $\alpha_0 = 0.1$. The results are presented in Figure 10. As expected,

(A) *Boxplot of ARIs for estimated session-level topics obtained via different initialisation schemes*



(B) *Barplot of estimated number of non-empty topics $K_\varnothing$ for different priors on $\boldsymbol{\lambda}$ (K_{max}-dimensional Dirichlet or GEM)*

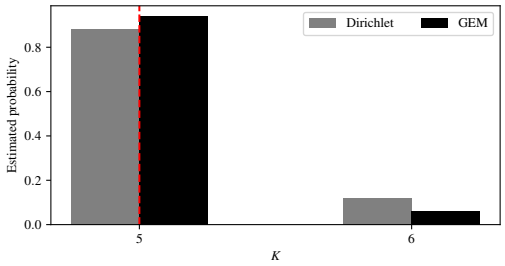
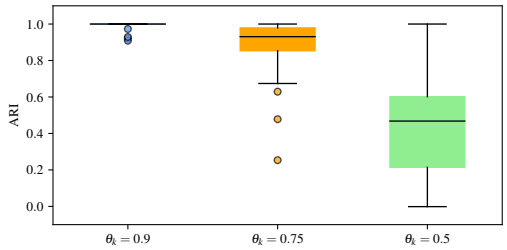


FIG 9: *Summary plots obtained from 50 simulated datasets from model (1), with* $|V| = 50,\ K = 5,\ D = 100,\ N_d = 10,\ M_{d,j} = 10,\ \boldsymbol{\lambda} = 1/K \cdot \mathbf{1}_K$, *and* $\boldsymbol{\eta} = 5 \cdot \mathbf{1}_K$. *The MCMC sampler is run for 25,000 iterations with 10,000 burn-in, setting* $K_{max} = 10$ *or a GEM prior for* $\boldsymbol{\lambda}$ *with* $\gamma = 0.1$.

(A) *Boxplot of ARIs for estimated session-level topics*



(B) *Barplot of estimated number of non-empty topics $K_\varnothing$*

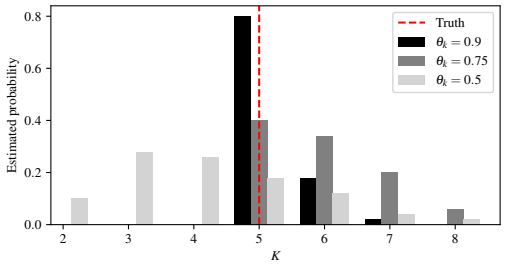
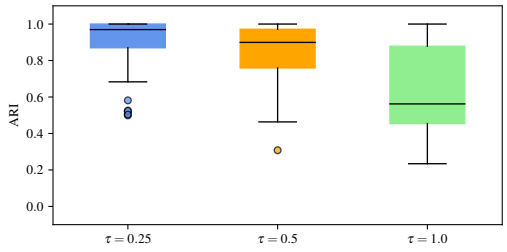


FIG 10: *Summary plots obtained from 50 simulated datasets from model (2), with* $|V| = 50,\ K = 5,\ D = 100,\ N_d = 10,\ M_{d,j} = 10,\ \boldsymbol{\lambda} = 1/K \cdot \mathbf{1}_K,\ \boldsymbol{\eta} = \mathbf{1}_K$, *using different values for* θ_k. *The MCMC sampler is run for 25,000 iterations with 10,000 burn-in, setting* $K_{max} = 10,\ \alpha_h = 0.9, \alpha_0 = 0.1$.

(A) *Boxplot of ARIs for estimated session-level topics*



(B) *Barplot of estimated number of non-empty topics $K_\varnothing$*

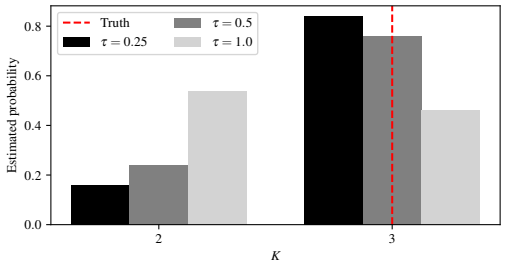


FIG 11: *Summary plots obtained from 50 simulated datasets from model (3), with* $|V| = 50,\ K = 3,\ H = 5,\ D = 100,\ N_d = 10,\ M_{d,j} = 10,\ \boldsymbol{\lambda} = 1/K \cdot \mathbf{1}_K,\ \boldsymbol{\eta} = \mathbf{1}_K$, *using different values for* $\boldsymbol{\tau} = \tau \mathbf{1}_H$. *The MCMC sampler is run for 25,000 iterations with 10,000 burn-in, setting* $K_{max} = 10,\ H_{max} = 10$.

Figure 10a shows that the ARI for the estimated session-level topics decreases when θ_k decreases. Furthermore, Figure 10b shows that estimation of the number of non-empty primary topics becomes increasingly imprecise when θ_k decreases, causing the drop in ARI observed in Figure 10a.

Third, datasets are simulated from model (3), using $\boldsymbol{\eta} = 0.1 \cdot \mathbf{1}_K$, $K = 3$, $H = 5$ and $\boldsymbol{\tau} = \tau \mathbf{1}_H$, for different values of τ. In this case, τ expresses how concentrated around a

peak the command-level topic distributions are. Small values of τ imply that the probability mass is mostly concentrated around one topic, whereas larger values correspond to more evenly distributed probability mass functions. In the MCMC sampler, the values $K_{\max} = 10$ and $H_{\max} = 10$ are used. Note that the task of recovering the session-level topics is much more complex than previous simulations, since for model (3), the session-level topic only controls the distribution of the command-level topics. Hence, the session-level topic must be estimated from the N_d command-level topics within each document, which are themselves estimated. This makes the inferential task substantially harder. Figure 11 displays the results: Figure 11a shows that the ARI for the estimated session-level topics tends to decrease when τ increases, and Figure 11b shows that estimates of the number of non-empty session-level topics are more precise when the value of τ used in the simulation is smaller. Notice that, since $\boldsymbol{\eta} = 0.1 \cdot \mathbf{1}_K$, the topic-specific word distributions have most of their mass concentrated around a small number of words, and the command-level topics are therefore easy to recover, with ARIs averaging above 0.99 across the three settings for τ shown in Figure 11.