Proceedings of the 55th Hawaii International Conference on System Sciences | 2022

TSM: Measuring the Enticement of Honeyfiles with Natural Language Processing

Roelien C. Timmer
UNSW Sydney
Data61, Cyber Security CRC
Sydney, Australia
r.timmer@unsw.edu.au

David Liebowitz
Penten
UNSW Sydney
Canberra, Australia
david.liebowitz@penten.com

Surya Nepal
Data61
Cyber Security CRC
Sydney, Australia
surya.nepal@data61.csiro.au

Salil S. Kanhere
UNSW Sydney
Cyber Security CRC
Sydney, Australia
salil.kanhere@unsw.edu.au

Abstract

Honeyfile deployment is a useful breach detection method in cyber deception that can also inform defenders about the intent and interests of intruders and malicious insiders. A key property of a honeyfile, enticement, is the extent to which the file can attract an intruder to interact with it. We introduce a novel metric, Topic Semantic Matching (TSM), which uses topic modelling to represent files in the repository and semantic matching in an embedding vector space to compare honeyfile text and topic words robustly. We also present a honeyfile corpus created with different Natural Language Processing (NLP) methods. Experiments show that TSM is effective in inter-corpus comparisons and is a promising tool to measure the enticement of honeyfiles. TSM is the first measure to use NLP techniques to quantify the enticement of honeyfile content that compares the essential topical content of local contexts to honeyfiles and is robust to paraphrasing.

1. Introduction

1.1. Honeypots for Cyber Deception

Cyber deception is increasingly being used in the defence of networks and information assets. A number of vendors are now advertising products that can detect, delay and mislead intruders and malicious insiders [1], generally with the creation and orchestration of honeypots. A honeypot is defined by Spitzner as *an information system resource whose value lies in unauthorized or illicit use of that resource* [2].

Honeypots commonly mimic devices such as file servers (the popular Thinkst Canary, for example[1]). They can be digital entities like document files, known as honeyfiles. A lot of valuable information is stored in documents such as financial reports, white papers, patent descriptions and contracts. Honeyfiles serve as digital traps just like other honeypots because they are fakes that mimic document files – they have no legitimate use, so any interaction with them is suspicious and suggests unauthorised access. Honeyfiles can also provide information about the interests and intent of the adversary. If we know the content of a honeyfile, the fact that an adversary is choosing to search for, open or exfiltrate that content tells us what they are trying to discover or steal. As the number of documents stored in repositories and knowledge management systems grows, so does the benefit of using honeyfiles to protect them against unauthorised access. For a honeyfile to successfully detect unauthorised access, it must attract the attention of an intruder [2]. The *enticement* of a honeyfile [3] is a measure of how well it does this job.

Engaging the attention of an adversary to achieve a deception outcome requires some understanding of the adversary's goals and perceptions. Generalised models of deception developed by Bell [4] and Whaley [5], based on observations from domains as diverse as the natural world, warfare and the practice of stage magic, describe identifying *channels* through which *ruses* can be communicated to the subject of the deception. In cyber deception, honeypots are typically "designed to seem as valuable as normal ones" [6] in order to entice the interaction that is essential to their use. Deception design can target the expected behaviour of an intruder, such as during the reconnaissance phase of a breach [7]. Some authors [8, 9] advocate designing deceptions based on a thorough understanding of the adversary to exploit bias in their perception. We describe the limited literature on honeyfile enticement below, and argue for an approach to quantifying the characteristics of honeyfile content that suits modern document repositories so that we can entice the appropriate honeyfile interactions. A measure is important for the practical use of honeyfiles. It informs design decisions around text content, particularly in the era of generative deep learning models, because we must ensure that honeyfiles are sufficiently enticing to operate effectively.

[1] see `https://canary.tools`

URI: https://hdl.handle.net/10125/79613
978-0-9981331-5-7

1.2. Honeyfiles

The first documented cyber defensive use of honeyfiles was Cliff Stoll's deployment of fake documents to trap an intruder on the Lawrence Berkeley National Laboratory networks in the 1980s [10, 11]. Stoll had been observing the intruder, and so could write honeyfiles with content and jargon to match their interests to entice the type and duration of interaction necessary to trace their location. Yuill *et al* [12] described a system for intrusion detection in which a user places a honeyfile on a file server with a mechanism that would trigger an alert when the file is opened. The user would then know they had been compromised if they received an alert. This innovative early concept honeyfile system relied on manually created files, with content chosen to be enticing to hackers by virtue of file names signalling that it contains information like credentials or financial data. Bowen *et al* [13] developed the Decoy Document Distributor, a system to automatically generate honeyfiles using templates for documents like tax returns and bank statements. A number of honeyfile properties were defined to describe their design, deployment and deceptive effects, including notions of how believable, enticing and conspicuous the files are. These properties were defined in thought experiments that supposed probabilities of outcomes should an intruder be faced with a decision involving a honeyfile. In [13] and a series of related papers [14, 15, 16, 17], honeyfile use was explored with a number of experiments testing these properties. Of particular interest, Ben Salem *et al* [14] investigated *enticingness* and *conspicuousness* in a study with 40 student subjects tasked with a scenario of theft from a desktop computer. They concluded that *conspicuousness comes first* for decoy effectiveness, that it is more important than enticingness in honeyfile access events.

1.3. Motivation

Whitham [18] developed a set of properties (similar to [13]) as design requirements for scalable, automated honeyfile generation, and followed up in [3] a method to automatically create enticing honeyfile content with NLP techniques. However, a sophisticated enticement based on NLP techniques is lacking. In current literature, honeyfile enticement is measured by simple word count. The enticement of a document is quantified by counting all the words common to a honeyfile and files in its local directory, and those shared by the honeyfile and the rest of the file system.

Two observations motivate our approach to enticement in this paper:

1) The experimental results obtained by Ben Salem *et al* [14], showing that the conspicuous placement of honeyfiles is more important than enticement, were derived from experiments on the local file system of a desktop computer. Users had to navigate the file system searching for files worth stealing, so placement had a significant impact. While the security of personal desktops remains a challenge, we believe that document theft is most damaging when it targets large scale repositories and knowledge management systems. Such systems store vast numbers of documents for governments and corporations, are indexed and searchable and present high-value targets for industrial espionage, state actors and other sophisticated adversaries. Searching for files in large repositories is likely to be through a search engine, not navigating a file system, making the text content of honeyfiles the primary means of promoting interaction with honeyfiles.

2) Whitham's [18] enticement measure uses words common to real files and honeyfiles. This approach is limited by insensitivity to paraphrasing and the use of synonyms – the same information expressed with different words with similar meaning would not count towards the enticement score.

1.4. Contributions

We use NLP methods to address these limitations by proposing a novel measure we call Topical Semantic Matching (TSM) as follows.

1) Topic modelling. We focus here on a scenario in which the adversary has access to a repository search engine and is looking for documents to steal. The indexed text content of the files is searched, so the enticement score must reflect how well honeyfile text represents the content of real files that may be a target. To generalise the idea of a local directory to subsets of a repository, we define the *local context* to be the set of documents files with a common theme or subject. We can define a local context in an indexed repository by the top N results returned by a set of search terms. Topics are the words representing the key themes appearing in all the documents in a corpus, so we compare honeyfiles to real files by comparing honeyfile text to the topics representing the real file corpus. This approach can be used with a local file system and placing a honeyfile in a local directory. Topic modelling is described in Section 2.1.

2) Semantic matching. The similarity between honeyfile and topics is quantified by the similarity of the *meanings* of the words. Semantic similarity compares words by their embeddings in a vector space where related words cluster together, and is described in Section 2.2.

We investigate TSM variants with the use of different topic modelling methods and aggregations of individual word similarities. Experiments on document corpora synthesised for this research show that TSM is a robust similarity measure when corpora derived from different category searches and domains are compared. This prepares the way to assess TSM as an enticement measure in an experimental setting.

Section 2 describes the NLP methods used and Section 3 introduces TSM. Section 4 describes our honeyfile corpus and experimental results. Our paper finishes with limitations and a discussion in Section 5 and a conclusion in Section 6.

2. Theory

The following section provides an overview of the four NLP methods used in this paper: topic modelling, semantic matching, doc2vec and text generation models. We limit the presentation to a brief summary and definition of the terminology that appear in subsequent sections, and refer the interested reader to accessible material for further information.

2.1. Topic Modelling

Topic modelling is a major subject of research in NLP that deals with extracting the main topics from texts. Topic models are particularly useful to extract hidden semantic structures. The output of topic models are clusters of similar words. The first topic models were created in the 1990s [19]. Topic models gained popularity when Blei, Ng, and Jordan introduced the Latent Dirichlet Allocation (LDA) model [20] in 2003. LDA is a hierarchical Bayesian model where every topic is modelled as a mixture of words, and a document is a mixture of topics. LDA has remained extremely popular, even as many other topic models have been published in recent years, such as the zero-shot cross-lingual [21], Stochastic Block Model [22], lda2vec [23] and embedded [24] topic models. A readable tutorial on LDA can be found in Darling[25].

In our comparative analysis, we use LDA [20] and SBM [22] topic models. We selected LDA because it is the most popular, and SBM as it has emerged recently with claims to better the performance of LDA.

2.2. Semantic Matching

Semantic matching, as used below, compares the meanings of words. Quantifying the semantic relationship between words uses vector representations that embed them in a high dimensional space – a semantic vector space. These embeddings can be created in a number of ways using words and their contexts. The context of a word is the words that appear just before and after it.

Early representations [26] used matrices to represent the co-occurrence of words and contexts, counting the number of times each word appears within each context. The rows and columns of these matrices represent the words and contexts as large, sparse vectors, or lower-dimensional, dense vectors after matrix factorisation [27]. Neural embeddings have emerged more recently, notably word2vec [28, 29], which computes embeddings using a neural network that predicts words or contexts (it has versions that do both). Similar embeddings can be derived from other approaches like GloVe [30] or FastText [31].

The vector representations we use to compute enticement scores do not depend on any embedding method, provided all the words are in the same semantic vector space. We compare the vector similarity of two words using the normalised inner product of the vectors. Let $\mathbf{x}$ and and $\mathbf{y}$ be (column) vectors of length f representing words u and v respectively:

$$\mathbf{x} = (x_1, x_2, ..., x_f) \text{ and } \mathbf{y} = (y_1, y_2, ..., y_f).$$

The inner product of x and y is the cosine similarity, and quantifies how similar the words u and v are in meaning:

$$\cos(\theta_{uv}) = \frac{x \cdot y}{\|x\| \|y\|} = \frac{\sum_{i=1}^{f} x_i y_i}{\sqrt{\sum_{i=1}^{f} x_i^2} \sqrt{\sum_{i=1}^{f} y_i^2}} \quad (1)$$

with $\|x\|$ and $\|y\|$ being the Euclidean norm which is the vector length. The cosine similarity can range from -1 to 1. When words are more similar, their vectors are closer to each other and the cosine similarity is closer to 1.

For ease of reference, we normalise the cosine similarity to the range $[0, 1]$ to yield a semantic similarity between words u and v:

$$S_{uv} = \frac{\cos(\theta_{uv}) + 1}{2} \quad (2)$$

Fig. 1 visualises why comparing words based on their semantic similarity is useful. The second sentence is a paraphrased version of the first. Standard word matching only detect two common words matches while semantic matching detects five.

2.3. Doc2vec

Sentences, paragraphs and whole documents can be represented by embeddings in a vector space. The embeddings represent the semantic content of the texts they represent, so that similarity measures may be applied. Doc2vec is a popular approach by Mikolov and Le [32], and uses a similar methodology to word2vec. For details on embedded representations and their use in NLP, see the tutorial by Goldberg [33]. A reasonable approach to enticement with semantic matching would be to compare doc2vec embeddings. To our knowledge, this has not been proposed in the literature, and we test it experimentally below. There are two different implementations of Doc2vec, the Distributed Memory Model of Paragraph Vectors (PV-DM) and the Distributed Bag-of-Words version of Paragraph Vector (PV-DBOW). Mikolov and Le showed based on experiments that PV-MD performs best on average [32]. In this paper, we, therefore, use the PV-DBOW implementation of Doc2vec. In our experiments , we test whether doc2vec embeddings can be used as the basis for a simple enticement measure. The results show that the TSM is a better enticement measure, probably because the doc2vec embeddings are strongly influenced by language other than the key topics in the documents.

2.4. Honeyfile Text Content

The text content of honeyfiles can be created manually, but this is time-consuming and prevents deployment at scale. Bowen *et al* [13] automated honeyfile creation with templates based on common documents. Whitham [3] used the text in files in a target directory to populate a honeyfile. This approach starts with Part-of-Speech (POS) tagging of n-grams in the reference text to assemble text fragments. Then a file from the target directory is used as a template by substituting text fragments in place, sampling from the assembled fragments while trying to match the POS-tagging of the n-gram fragments in the template.

Recent developments in language models have given rise to enabling technology that can mimic human text remarkably well. Such automatic text generation received lots of attention in 2018 when the Generative Pre-trained Transformer, better known as GPT, was released by OpenAI [34]. An improved model, GPT-2 [35], was released soon after in 2019. In 2020 GPT-3 followed [36]. The number of parameters in the model increased dramatically over these three different versions, from 117 million parameters to 1.5 billion parameters and eventually to 175 billion parameters. There are now specialised text generation models, for example, a model that generates coherent and cohesive long-form texts [37]. Recent publications have focused specifically on fake text or fake code generation for honeyfiles [38, 39, 40]. A cyber security company [anon] has a product that creates and manages the lifecycle of honeyfiles to protect document repositories [anon]. We use honeyfiles generated by this technology to test the proposed measure, as described in Section 4. This technology supports training the text generation model with three approaches: a POS-tagged substitution model, a similar model that uses dependency parsed tokens (DPT) instead of POS-tagged n-grams, and a GPT-2 model that can be fine-tuned on a corpus. It supports using the standard Lorem Ipsum dummy text instead of a trained model. Lorem Ipsum is a pseudo-Latin text often used as a placeholder [41].

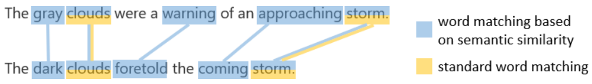

The gray clouds were a warning of an approaching storm.
The dark clouds foretold the coming storm.
word matching based on semantic similarity
standard word matching

Figure 1. The difference between word matching based on semantic similarity and the common words.

The Lorem Ipsum text should not match any local context. Because the POS-tagged and DPT train a text substitution model using fragments from the local context, the honeyfile contains real text, but in a garbled form. GPT-2 mode uses a model fine-tuned on the local context corpus. It contains text semantically similar to the local context due to fine-tuning and is usually more realistic than the other methods.

3. Topical Semantic Matching

We propose the Topical Semantic Matching (TSM) enticement score for honeyfiles. This novel enticement measure is based on topic modelling to capture the topics contained in the local context and semantic matching to measure the similarity between the honeyfile and topics. TSM can be calculated with a topic model of choice.

We refer to the honeyfile text as h and the local context text as l. The honeyfile h is a set of N_K words:

$$h = \{u_1, u_2, \ldots, u_{N_K}\} \tag{3}$$

The local context l is composed of M words from all the files in the local context:

$$l = \{w_1, w_2, \ldots, w_M\} \tag{4}$$

We preprocess the honeyfile h and local context l by removing stop words, applying named entity recognition (NER) and lemmatisation. These three steps are all common NLP preprocessing practices, giving h' and l', with $N_{K'}$ and $N_{M'}$ words respectively. Then we extract the topics from L' in the form of a set of N_T topic words to give $t = \{v_1, v_2, \ldots, v_{N_T}\}$.

Algorithm 1: TSM Score

```
Input: honeyfile $h$, local context $l$, threshold $\delta$
Output: enticement score $E$
$l'$, $h'$ : preprocess $l$ and $h$ by removing stop words,
  applying lemmatisation and named entity
  recognition
$t$ : extract the topics from $l'$
$\mathsf{L}, \mathsf{H}$ : represent each word of $l'$ and $t$ by their
  embedding
$\mathsf{S}_{\mathsf{HT}}$ : normalise all columns of $\mathsf{H}$ and $\mathsf{T}$ to length 1,
  calculate the matrix product $\mathsf{HT}^T$ and scale to the
  range[0, 1] using Equation 2.
if thresholding method then
    $E = \sum s_i/(N_{K'} * N_T)$ for $s_i \in \mathsf{S}_{\mathsf{HT}}, s_i \geq \delta,$
      $0 \leq \delta \leq 1$
else
    use the averaging method
    $E = \sum(\mathsf{S}_{\mathsf{HT}})/(N_{K'} * N_T)$
```

To extract the topics, a topic model of choice can be used. For our comparative analysis we selected LDA because of its popularity, and SBM as it has emerged recently with claims to better performance than LDA.

Next, we want to quantify the similarity in meaning between the honeyfile and the local context as represented by the topic words. To do this, we construct matrices H and T from sets h' and t by representing each word in h' and t by their embedding vectors:

$$\mathsf{H} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_{N'_K}\right] \text{ and } \mathsf{T} = \left[\mathbf{y}_1, \ldots, \mathbf{y}_{N_T}\right]$$

where we use $\mathbf{x}$ for honeyfile word vectors and $\mathbf{y}$ for topic word vectors.

This representation has computational advantages when computing similarity scores by using efficient linear algebra implementations available in most numerical libraries [42]. To do this we normalise the columns of H and T to length 1, calculate the matrix product $\mathsf{HT}^\top$ and scale each element to the range $[0, 1]$ using Equation 2 to give S_{HT}, the semantic similarity between all word pairs from h' and t.

We explore two methods to aggregate these similarities into a score: averaging all similarity scores between the topics and the honeyfiles words, and thresholding before averaging. As data thieves often search for specific keywords, we expect that a high threshold is most suitable to measure honeyfile enticement. This gives the final TSM enticement score E of the honeyfile h with respect to the local context. Fig. 2 shows a visual representation of the TSM measure and Algorithm 1 outlines the pseudo-code [2].

4. Comparative Analysis

In this section, we explain how we created our honeyfile corpus and we show that the TSM measure with threshold performs best.

4.1. Honeyfile and Local Context Generation

We could not find a suitable corpus that contains honeyfiles and local context files for our experiments. We thus generated one ourselves. We scraped files from the Internet that serve as our local context and as input for the generation of the honeyfiles. We selected technical files as hackers are often interested in intellectual property and other technical documents. For this paper, the technical files are academic papers from different disciplines and a set of official customs notices published by the Australian Government.

The experimental corpora contain four different types of files of which three are academic papers with author keywords[3] theater, computer architecture and plants. The fourth type of file is Australian customs notices[4].

Table 1 shows the number of files per category. In this paper, a local context consists of 5, 10 or 20 different local context files from the same category. For the customs notices, we selected files for a local context based on release dates. For example, for a local context of size 10, the first file in the custom local context is 1996-01 and the last is 1996-10.

Each category of local context files also serves as the input to generate honeyfiles. Four honeyfile generation methods are used: GPT-2 based, POS-tagging, DPT or replacement with Lorem Ipsum text. For all the four

Table 1. The number of web scraped local files also serve as the training corpus for the honeyfiles.

Category Local Files	Number
Australian customs notices	1460
Papers about 'theater'	100
Papers about 'computer architecture'	100
Papers about 'plants'	140
Total	1800

[a] Every local contexts consist of 5, 10 or 20 files of a category.

[2] The implementation of TSM can be found on https://github.com/RoelTim/tsm-honeyfile-nlp-enticement.

[3] These papers are downloaded from Web of Science. We randomly selected papers with the corresponding author keyword and English as the main language. The author keyword 'plants' refers to biological plants and not to industrial plants.

[4] Australian Custom Notices are online at https://www.abf.gov.au/help-and-support/notices/australian-customs-notices. We downloaded files in the date range 1996-01 to 2020-39.

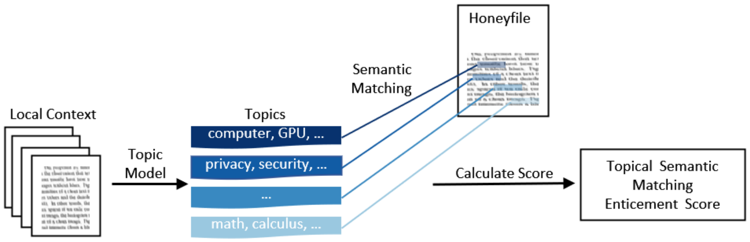
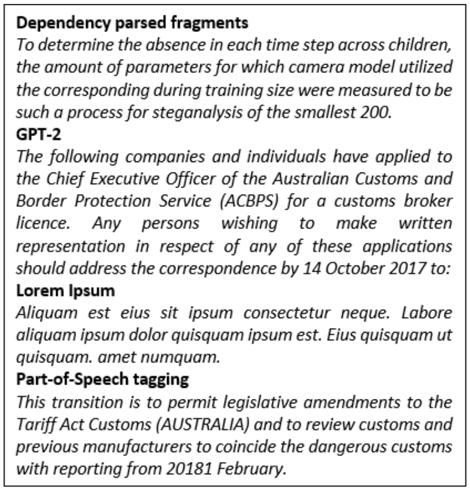


Figure 2. To calculate TSM, a topic model extracts the topics from the local context. TSM is the weighted average of the semantic similarities between these topics and the honeyfile words above a certain threshold.



Dependency parsed fragments

To determine the absence in each time step across children, the amount of parameters for which camera model utilized the corresponding during training size were measured to be such a process for steganalysis of the smallest 200.

GPT-2

The following companies and individuals have applied to the Chief Executive Officer of the Australian Customs and Border Protection Service (ACBPS) for a customs broker licence. Any persons wishing to make written representation in respect of any of these applications should address the correspondence by 14 October 2017 to:

Lorem Ipsum

Aliquam est eius sit ipsum consectetur neque. Labore aliquam ipsum dolor quisquam ipsum est. Eius quisquam ut quisquam. amet numquam.

Part-of-Speech tagging

This transition is to permit legislative amendments to the Tariff Act Customs (AUSTRALIA) and to review customs and previous manufacturers to coincide the dangerous customs with reporting from 20181 February.

Figure 3. Text snippets from the honeyfiles.

methods we started with a template[5]. The honeyfiles generated mimic the layout of these templates.

The GPT-2 based honeyfiles were generated by fine-tuning the pre-trained GPT-2 medium model on the four different local context corpora. For example, a 'computer architecture' honeyfile was generated by fine-tuning on the 'computer architecture' corpus. The POS-tagging honeyfiles were generated by replacing the words in the template with another word that has the same POS-tagging. The DPT files were generated by imitating the DPT structure of the template. The Lorem Ipsum honeyfiles were generated by replacing the text in the templates with Lorem Ipsum. Fig. 3 shows text snippets of the different types of honeyfiles generated.

We expect that a honeyfile h generated based on a corpus has a high enticement score when compared to a local context file from that corpus. For example, a honeyfile that was trained on 'theater' files should have a high enticement score if it is compared with a local context from the 'theater' category.

4.2. Selecting the Best Enticement Measure

In this section, we show that TSM with thresholding is best in measuring enticement and show the results on the corpus we created.

Fig. 4 shows that TSM with thresholding has a higher enticement score for honeyfiles matched with local contexts from the same category. This heatmap shows that the doc2vec and TSM without thresholding perform poorly. Intuitively this makes sense as TSM with thresholding only takes into account words that have a high semantic similarity with the main topics. TSM with thresholding filters out noise as it ignores words that have a low semantic similarity. The three diagonals on the top left corner show that the enticement scores is higher between honeyfiles and local contexts that correspond to the same corpus. For example, a honeyfile that was generated based on the 'computer architecture' corpus matched with a local context based

[5] We have eight different templates which are derived from the local context files. The eight templates are: 1996-01, 1996-17, 2018-02, 2019-40, 2020-39, 08552374, 0021989420918654 and wild-useful-herbs-of-aktobe-region-western-kazakhstan.

Table 2. Number of Honeyfiles Generated with Different Generation Methods and Based on Different Corpora.

	Honeyfile Training Corpus				
	Australian custom notices	Theater	Computer architecture	Plants	Total
GPT-2	103	25	25	25	178
Lorem Ipsum	N/A	N/A	N/A	N/A	160
POS-tagging	100	20	20	20	160
Dependency Parsed Tokens	100	20	20	20	160
Total	303	65	65	65	498

on the ‘computer architecture’ corpus has a relatively high enticement score. The topic model used for the TSM scores is LDA with five topics each consisting of ten words. Each local context consists of 10 files. Fig. 5 shows the distribution of the enticement scores of TSM with a threshold of 0.9 and 10 files per local context.

We include doc2vec in the comparison as doc2vec takes into account all words of a document [32], while TSM only extracts the main topics. We trained the doc2vec model on the preprocessed Australian custom notices and the academic articles. Next, we trained our doc2vec model on a data set similar to the local context l. The doc2vec model can use the Distributed Memory Model of Paragraph Vectors (PV-DM) or the Distributed Bag of Words version of Paragraph Vector (PV-DBOW)[6]. The first (i.e., PV-DM) generally gives better results, but requires a longer training time. After training the doc2vec model we extract the embedding of the honeyfile h and the local context l. As a final step, to calculate the enticement score, we calculate the cosine similarity between these two vectors.

Fig. 4 shows that for the doc2vec measure the enticement scores are not necessarily higher between honeyfiles and local contexts that correspond to the same corpus. As expected, it is better to focus on the main topics than on the whole document as with doc2vec.

All these results are consistent over the three main honeyfile generation methods which are based on GPT-2, DPT and POS-tagging. The honeyfiles containing Lorem Ipsum text have enticement scores of or close to zero. This is in line with our expectations, as Lorem Ipsum text is not related to any of the corpora.

The next experiments use default variables unless stated otherwise. The local context size is 10 and the topic model is LDA[7]. For LDA we select five topics each consisting of ten words. We aggregate the GPT-2, DPT and the POS-tagging honeyfiles. We leave out the Lorem Ipsum honeyfiles as we consider them weak honeyfiles. All the heatmaps show the 50th percentile of the enticement scores.

4.3. Experimenting with Thresholds

We experimented with several TSM similarity thresholds. Fig. 6 shows that a high threshold of 0.9 yields a good result. This is not surprising as only very similar words get matched with a high semantic similarity. This means that irrelevant words are ignored, and that enticement is well represented by distances between the most similar topics and honeyfile words. Instead of aggregating the semantic similarities that surpass a certain threshold, we can aggregate only the highest similarities. Fig. 7 shows that selecting the top 0.5% of the highest semantic similarities leads to good results. Intuitively this makes sense, as in line with the thresholding method, we only capture the topics that

[6] Our parameters are max_epochs = 100, vec_size = 20, alpha = 0.025 and embedding model = Distributed Memory (PV-DM).

[7] We use the GENSIM LDA model: https://radimrehurek.com/gensim/models/ldamodel.html

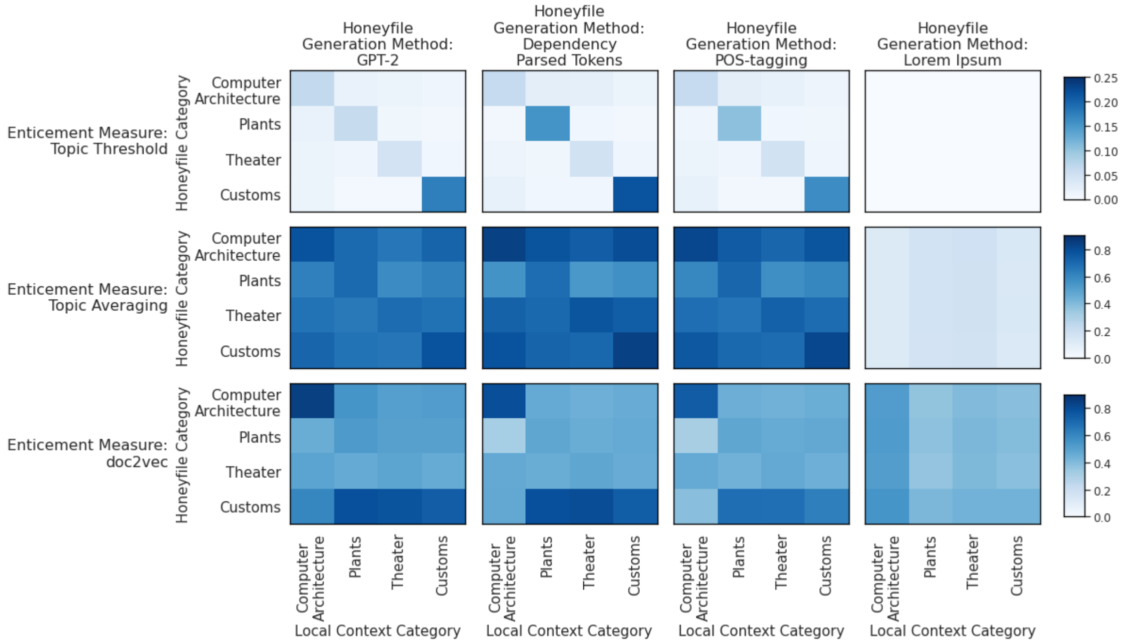


Figure 4. The 50th percentile of the enticement scores between the honeyfiles matched with local context files.

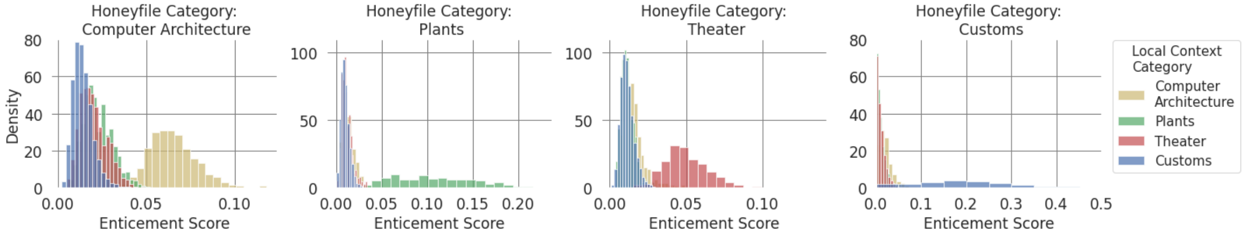


Figure 5. The distribution of the TSM enticement score.

have a high similarity with the text in the honeyfile.

4.4. Experimenting with Topics Models

TSM does not depend on any particular topic model. Fig. 8 shows that the results of the LDA and SBM topic models are comparable. For the SBM topic model we select the default level $l = 1$ and the number of words $n = 50$[8]. Next, we select the 50 words that contribute most to the $l = 1$ topics.

We experimented with replacing the topic models by selecting the 50 most common words of the preprocessed local context files. Fig. 8 shows that the results of selecting the 50 most common words are similar to applying a topic model.

The biggest advantage of selecting the most common words is the low computational cost. Selecting the most common words is faster than running a topic model, although in general better results can be expected using topic modelling.

4.5. Experimenting with Local Context Sizes

In practice, the local contexts vary in size. Therefore, we experimented with different local context sizes of 5, 10 and 20 files. Fig. 9 shows that the performance of TSM is similar when the local context size varies. The heatmaps shows that TSM results are almost identical for the different local context sizes.

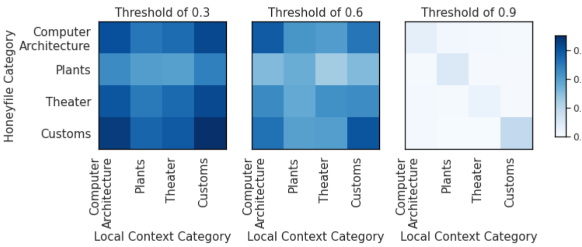


Figure 6. TSM works best with a high threshold.

5. Limitations and Discussion

TSM is proposed as a metric to measure the enticement of the text content in a honeyfile. It is aimed at a context in which honeyfiles protect real documents in a repository accessible by search. We argue that the honeyfile text should be semantically similar to the topics represented in the real files so that they are as likely as the real documents to be returned by meaningful search terms. This is necessary for the honeyfiles to work, since interaction is the signal that drives the defensive benefits.

Enticement as a measure should reflect how well the honeyfile content attracts the attention of intruders. We show experimentally that TSM correctly reflects the semantic similarity of honeyfiles and the topics of text corpora. This suggests that it is a plausible candidate for an effective enticement measure. While there is a growing body of experimental work in cyber deception, such as the Tularosa Study [43, 44], it does not address the specific interactions associated with honeyfiles or text content. We intend to test the effectiveness of TSM as an enticement metric in human trials with text-based interactions.

TSM, as implemented here, cannot distinguish between homonyms. For example, the word ‘good’ is used in Australian Customs notices to mean a product while in other documents it is used as an adjective. It could potentially be improved using part-of-speech or dependency parsed word embeddings [45, 46].

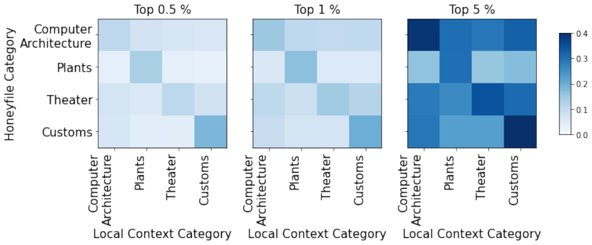


Figure 7. Aggregating the top 0.5% semantic similarities give similar results to TSM with threshold.

[8] Link to the code of the SBM topic model: `https://github.com/martingerlach/hSBM_Topicmodel`

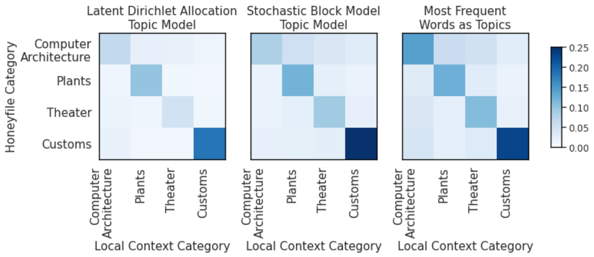


Figure 8. TSM score are comparable for different topic models and when the top 50 words are used.

A limitation likely to be seen in practice with small corpora is the constrained ability of topic models to extract reliable topic word distributions. Approaches such as topic cropping [47] might alleviate this somewhat, but small training corpora will be challenging from a text generation perspective as well.

A general limitation of honeyfile research is the relatively small literature and absence of standard data sets. Thus, there is no golden data set that we can test our TSM measure on. We anticipate that our future user study will provide a data set that can be used for this purpose.

6. Conclusion

In this paper, we used an NLP-based approach we call Topic Semantic Matching to develop a measure of the enticement of honeyfiles. TSM compares words in the honeyfiles to topics representing the real documents they protect. We show experimentally that TSM with a high threshold performs well at comparing the semantic content of honeyfiles to the corpora that its generative models trained on relative to comparable corpora. Similar results are achieved with alternative topic models.

The key advantage of TSM is that it is robust to paraphrasing and the use of synonyms through the use of semantic matching. The existing common word count measure accounts for words that are exactly the same, while TSM accounts for similar meanings of words.

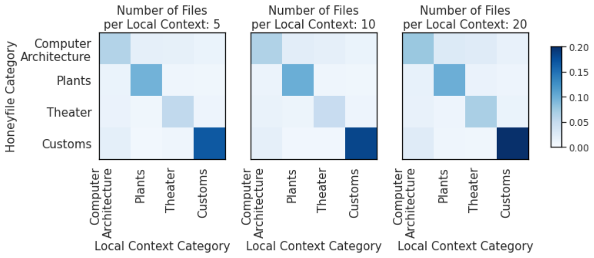


Figure 9. The TSM scores are comparable when the local contexts differ in size.

TSM is well suited to contexts in which real documents and honeyfiles are stored in a repository accessed via search terms, and we believe it is a promising candidate to evaluate as a practical enticement measure. Currently, we are working on a study that tests the TSM measure and other metrics on human subjects. We plan to also investigate quantitative measures of other honeyfile properties, such as realism and the presence of sensitive information. Realism, presence of sensitive information and enticement are often in conflict with each other. For example, a honeyfile that is almost a duplicate of a document that we want to protect is considered highly enticing and realistic but is also more likely to contain sensitive information.

References

[1] S. Underwood. (2020) Tactical deceit. [Online]. Available: https://cacm.acm.org/news/245079-tactical-deceit

[2] L. Spitzner, “Honeypots: Catching the insider threat,” in *19th Annu. Computer Security Applicat. Conf., Proc.* IEEE, 2003, pp. 170–179.

[3] B. Whitham, “Automating the generation of enticing text content for high-interaction honeyfiles,” in *Proc. the 50th Hawaii Int. Conf. System Sciences*, 2017.

[4] J. B. Bell, “Toward a theory of deception,” *Int. J. of Intell. and CounterIntell.*, vol. 16, no. 2, pp. 244–279, 2003.

[5] B. Whaley, “Toward a general theory of deception,” *J. of Strategic Stud.*, vol. 5, no. 1, pp. 178–192, 1982.

[6] S. Rauti and V. Leppänen, “A survey on fake entities as a method to detect and monitor malicious activity,” in *Parallel, Distributed and Network-based Processing 25th Euromicro Int. Conf.*

[7] D. Fraunholz and H. D. Schotten, “Defending web servers with feints, distraction and obfuscation,” in *Proc. Int. Conf. Computing, Networking and Communications*.

[8] M. H. Almeshekah and E. H. Spafford, “Cyber security deception,” in *Cyber deception*. Springer, 2016, pp. 23–50.

[9] C. Wang and Z. Lu, “Cyber deception: Overview and the road ahead,” *IEEE Security & Privacy*, vol. 16, no. 2, pp. 80–85, 2018.

[10] C. Stoll, “Stalking the wily hacker,” *Communications of the ACM*, vol. 31, no. 5, pp. 484–497, 1988.

[11] ——, *The Cuckoo’s Egg: Tracking a Spy through the Maze of Computer Espionage*. Simon and Schuster, 2005.

[12] J. Yuill, M. Zappe, D. Denning, and F. Feer, “Honeyfiles: deceptive files for intrusion detection,” in *Proc. the 5th Annu. SMC Information Assurance Workshop*.

[13] B. M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo, “Baiting inside attackers using decoy documents.” in *SecureComm*, vol. 19. Springer, 2009, pp. 51–70.

[14] M. B. Salem and S. J. Stolfo, "Decoy document deployment for effective masquerade attack detection," in *Int. Conf. Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2011, pp. 35–54.

[15] J. Voris, N. Boggs, and S. J. Stolfo, "Lost in translation: Improving decoy documents via automated translation," in *Security and Privacy Workshops, IEEE Symp. on*, 2012, pp. 129–133.

[16] J. Voris, J. Jermyn, A. D. Keromytis, and S. J. Stolfo, "Bait and snitch: Defending computer systems with decoys," in *Proc. the cyber infrastructure protection Conf., Strategic Stud. Institute, September*, 2013.

[17] J. Voris, J. Jermyn, N. Boggs, and S. Stolfo, "Fox in the trap: thwarting masqueraders via automated decoy document deployment," in *Proc. the 8th European Workshop on System Security*. ACM, 2015, p. 3.

[18] B. Whitham, "Design requirements for generating deceptive content to protect document repositories," in *Proc. the 16th Australian Inform. Warfare Conf., Security Research Institute, Edith Cowan University*, 2014.

[19] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala, "Latent semantic indexing: A probabilistic analysis." New York, NY, USA: Association for Computing Machinery, 1998.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. of Mach. Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[21] F. Bianchi, S. Terragni, D. Hovy, D. Nozza, and E. Fersini, "Cross-lingual contextualized topic models with zero-shot learning," *arXiv preprint arXiv:2004.07737*, 2020.

[22] M. Gerlach, T. P. Peixoto, and E. G. Altmann, "A network approach to topic models," *Science advances*, vol. 4, no. 7, p. eaaq1360, 2018.

[23] C. E. Moody, "Mixing dirichlet topic models and word embeddings to make lda2vec," *arXiv preprint arXiv:1605.02019*, 2016.

[24] A. B. Dieng, F. J. Ruiz, and D. M. Blei, "Topic modeling in embedding spaces," *Trans. of the Association for Comput. Linguistics*, vol. 8, pp. 439–453, 2020.

[25] W. M. Darling, "A theoretical and practical implementation tutorial on topic modeling and gibbs sampling," in *Proc. the 49th Annu. meeting Assoc. for Comput. Linguistics: Human language Technol.*, 2011, pp. 642–647.

[26] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Int. Res.*, vol. 37, no. 1, pp. 141–188, Jan. 2010.

[27] Y. Goldberg, O. Levy, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Trans. of the Association for Comput. Linguistics*, vol. 3, 2015.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[29] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[30] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[31] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. of the Association for Comput. Linguistics*, vol. 5, pp. 135–146, 2017.

[32] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. the 31st Int. Conf. Machine Learning*.

[33] Y. Goldberg, "A primer on neural network models for natural language processing," *J. of Artificial Intell. Research*, vol. 57, pp. 345–420, 2016.

[34] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[36] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[37] W. S. Cho, P. Zhang, Y. Zhang, X. Li, M. Galley, C. Brockett, M. Wang, and J. Gao, "Towards coherent and cohesive long-form text generation," *arXiv preprint arXiv:1811.00511*, 2018.

[38] P. Karuna, H. Purohit, O. Uzuner, S. Jajodia, and R. Ganesan, "Enhancing cohesion and coherence of fake text to improve believability for deceiving cyber attackers," in *Proc. the 1st Int. Workshop on Language Cognition and Computational Models*, 2018, pp. 31–40.

[39] P. Karuna, H. Purohit, R. Ganesan, and S. Jajodia, "Generating hard to comprehend fake documents for defensive cyber deception," *IEEE Intell. Syst.*, vol. 33, no. 5, pp. 16–25, 2018.

[40] D. Nguyen, D. Liebowitz, S. Nepal, and S. Kanhere, "Honeycode: Automating deceptive software repositories with deep generative models," in *Proc. the 54th Hawaii Int. Conf. Syst. Sci.*, 2021, p. 6945.

[41] O. Team *et al.*, "Why people use lorem ipsum to represent dummy text? the research of loerem ipsum," *J. Of Educ.*, vol. 1, no. 1, pp. 11–16, 2019.

[42] K. Fatahalian, J. Sugerman, and P. Hanrahan, "Understanding the efficiency of gpu algorithms for matrix-matrix multiplication," in *Proc. the ACM Conf. Graphics hardware*, 2004, pp. 133–137.

[43] K. Ferguson-Walter, T. Shade, A. Rogers, M. C. S. Trumbo, K. S. Nauer, K. M. Divis, A. Jones, A. Combs, and R. G. Abbott, "The tularosa study: An experimental design and implementation to quantify the effectiveness of cyber deception." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2018.

[44] K. J. Ferguson-Walter, M. M. Major, C. K. Johnson, and D. H. Muhleman, "Examining the efficacy of decoy-based and psychological cyber deception."

[45] D. Suleiman and A. A. Awajan, "Using part of speech tagging for improving word2vec model," in *IEEE 2nd Int. Conf. new Trends in Comput. Sciences*, 2019, pp. 1–7.

[46] O. Levy and Y. Goldberg, "Dependency-based word embeddings." in *ACL (2)*, 2014, pp. 302–308.

[47] N. K. Tran, S. Zerr, K. Bischoff, C. Niederée, and R. Krestel, "Topic cropping: Leveraging latent topics for the analysis of small corpora," in *Int. Conf. Theory and Practice of Digit. Libraries*. Springer, 2013, pp. 297–308.