Utrecht University

Masters Thesis

Embedded Clustering by Intent: an Active Network Traffic Clustering method

Author
Joël Plantinga

First Supervisor
Dr. Ing. H. G. Krempl
Utrecht University

Second Supervisor
Dr. I. R. Karnstedt-Hulpus
Utrecht University

Daily Supervisor
F. Miedema MSc
TU Delft

Daily Supervisor
J. Ranzijn MSc

A thesis submitted in fulfillment of the requirements for the degree of Business Informatics

in the

Department of Information and Computing Sciences

April 22, 2024

Acknowledgement

First of all, I would like to thank my supervisor Dr. Ing. H. G. Krempl for his support and guidance during this project. In particular, he helped me a lot to choose a direction for the project when I was struggling to find a good angle.

I would also like to thank my second supervisor, Dr. I. R. Karnstedt-Hulpus, for all the feedback she provided throughout this project. She went beyond the expectations of a second supervisor. She helped me shape the project by providing ideas for the evaluation and guidance for the structure.

This thesis project would not have been possible without my daily supervisors Jorn and Fieke. At the beginning of this project I had almost no knowledge of the cyber domain. They not only helped me understand the domain, but also made it a very enjoyable experience.

Furthermore, I would like to thank the people at Team High Tech Crime from the Dutch National Police. They helped me a lot by sharing their experience with network traffic analysis. It was a pleasure to work with so many talented people.

Last but not least, I want to express my gratitude to my girlfriend, family, and friends. Having them around me is invaluable.

Abstract

Gathered data sources hold increasing significance for law enforcement agencies to find evidence for criminal investigations. Network traffic stands out as one such data source, offering detailed insights into individuals' behaviors and interests. Given the potential size of intercepted network traffic, clustering methods are required to find what activities were performed in the data; methods that do not yet exist. Unlike text data, internet sessions lack inherent coherence and often include noise from unrelated traffic. Traditional unsupervised methods may yield suboptimal results due to this noise, while supervised methods are infeasible due to the absence of labeled data. To address these challenges, we proposes an adaptation of the 'Clustering By Intent' (CBI) technique to network traffic, named Embedded Clustering By Intent (ECBI). ECBI uses Word2vec embeddings to generate queries for identifying synonymous features in internet traffic, enabling clustering based on actual activities. The proposed method is validated using specifically sampled internet traffic from the Dutch National Police. In an evaluation consisting of expert interviews and technical experiments, ECBI is compared against CBI and Embedded Topic Modelling. Notably, ECBI clusters contain sessions from a significantly larger number of entities, indicating a focus on activities rather than specifics of an individual entity. Furthermore, we show that Word2vec can effectively be applied on network traffic.

Contents

1 Introduction 5
1.1 Research questions . . . 6
1.2 Contributions . . . 8

2 Background & related work 9
2.1 Network traffic analysis . . . 9
2.1.1 Network traffic . . . 9
2.1.2 Network forensics . . . 11
2.1.3 Machine learning approaches for analysing network traffic 13
2.2 Data mining techniques . . . 14
2.2.1 Bi-clustering & bipartite graphs . . . 14
2.2.2 Clustering high-dimensional data . . . 15
2.2.3 Semi-supervised clustering . . . 16
2.2.4 (Inter)active clustering . . . 17
2.2.5 Feature embedding . . . 18
2.2.6 Topic modelling . . . 19

3 Embedded Clustering by Intent 20
3.1 Data Model & Notation . . . 20
3.2 Problem statement . . . 22
3.3 Methods . . . 23
3.3.1 Overview of the method . . . 23
3.3.2 Research question 1 . . . 23
3.3.3 Research question 2 . . . 26
3.3.4 Research question 3 . . . 28

4 Evaluation of Connection Embeddings 31
4.1 Data . . . 31
4.2 Connection embedding experiment . . . 32
4.2.1 Setup experiment . . . 32
4.2.2 Results . . . 34

5 Evaluation of Embedded Clustering by Intent 36
5.1 Evaluation protocol . . . 36
5.2 Expert interviews for the final application . . . 37
5.2.1 Setup expert interviews . . . 37
5.2.2 Results . . . 39

5.3 Technical experiments for the final application 41
5.3.1 Setup experiment 41
5.3.2 Results 43

6 Discussion **46**

7 Conclusion **50**

A Full technical experiments results **58**

Chapter 1

Introduction

As life increasingly transitions into the digital realm, so does criminal behavior. Not only are victims more often targeted by cybercrime, but online traces left by criminals are also more important for finding evidence. Intercepted network traffic is one of the most important data sources available as it gives a detailed image of how people operate and what they are interested in. Network traffic consists of network packets: the basic unit of data transmission in computer networks. A packet consists of a payload and a header containing control information, including the source IP and destination IP [61]. As the quantity of the network traffic can be very large, computational methods are required to acquire insights from the data. We aim to explore the possibility to cluster internet traffic based on performed activity. This is achieved by examining internet sessions - uninterrupted periods of time during which the internet has been used - of an entity in the data.

To the best of the author's knowledge, no research exists in which network traffic was studied with the intention to cluster multi-entity data to find the performed activity. This is ultimately associated with the delicate nature of network traffic, as it contains highly personal information and is safeguarded by data protection regulations such as the General Data Protection Regulation (GDPR). The network traffic consists of connections, data exchanges between the entity and some destination, that occur in sequence and can be regarded to as discrete objects. For that reason, the comparison with text - where internet sessions represent documents and connections represent words - is made and techniques from the text mining domain are applied.

However, while similarities with text exist, notable differences in the characteristics of the data require adaptions to be made to obtain sensible results. A difference with text is that internet sessions are not created with the intention to inform or convey a reader, something that requires coherence. Instead, besides containing traffic that is relevant for the clustering objective, it might as well contain traffic related to the operating system. For this reason, there is a considerable amount of noise in the data. Relying on a fully unsupervised method would hence be optimistic as all kind of inferences could be made. Yet, supervised methods cannot be used as labelled data does not exist due to the combination of data size, complexity of labelling, and absence of open source data.

By incorporating the expertise of domain experts, it is possible to guide the model towards better results. While semi-supervised learning methods - that accept *must link* or *cannot link* constraints - aim to accept this feedback, it is not feasible to provide this for the web sessions. As domain experts build understanding on features - e.g., the domain names that connections are related to - it is hard to provide relatedness about individual sessions. *Clustering By Intent* (CBI) [25] is a text mining technique that allows experts to give feedback on a set of texts - i.e. a cluster - that have been put together given one or more common words that are unique to them. This gives the experts an indication of why certain sessions are grouped together and allows them to provide feedback on this specification.

An important aspect of network traffic, however, is that the same action can often be performed in many different ways. For that reason, traffic that indicates the same activity - e.g., traffic to competing travel agency websites - might end up to be seen as different. While this phenomenon - i.e., synonyms - exists in textual data as well, it is significantly more prominent in network traffic. In the english language, there are around 10[1] synonyms for each word. Looking at domains, popular websites might already have more than 10 different top-level domains and most of these websites have direct competitors providing a similar service. This thesis project aims to adapt CBI to network traffic by implementing Embedded Clustering By Intent (ECBI). This extends CBI by generating queries using Word2vec embeddings to find synonym features. These extended queries allow the model to find clusters based on the actual activity rather than a particular provider of that activity, making it more robust. Given the additional challenge of requiring feedback from analysts, visualization methods have been implemented to help them make informed decisions.

We validated the method on specifically sampled internet traffic from the Dutch National Police. Using the clustering results, the police aim to identify the interests of the entities, assuming that these can be perceived through the activities in which the entity is involved. The evaluation, consisting of expert interviews and a series of technical experiments comparing ECBI against CBI and Embedded Topic Modeling, found that the technique could be a promising tool for network traffic analysis. Clusters created by ECBI tend to include sessions of significantly more entities that clusters from CBI. This seems to indicate that clusters are created around activities rather than individual set-up's of individual entities.

1.1 Research questions

In order to find meaningful clusters that capture the activity performed in a session the following assumption is necessary and thus made: *activities an entity is involved in are perceived through the network traffic produced by the entity.*

[1] This is the average number of synonyms for the 5000 most frequent words in the Corpus of Contemporary American English (COCA). Synonyms are determined by using the *WordNet* [64] dataset.

With this assumption, it would be possible to programmatically find clusters of sessions using its attributes. To find a method that yields meaningful clusters, it is important to consider the characteristics that distinguish network traffic from textual data. As mentioned before, unlike most texts, internet sessions are not created with a reader in mind. For that reason, even more so than in textual documents, internet sessions can be clustered in many different ways. An internet session can contain traffic related to the operating system, professional or personal usage. For this reason, in comparison to clustering academic authors, the clustering exercise would involve clustering them not only on their articles. Instead, it would have to do so on randomly concatenated documents consisting of their articles, personal messages, and correspondence with their bank. For this reason, it is important that a domain expert can provide external knowledge and guide the model towards more meaningful results: clusters that have the relevant performed activity of the entity in common. The following research question is proposed:

How to actively cluster network traffic based on an analyst's intent?

By *bucketing* - abstracting connections to one or more features - individual connections, each internet session and its connections can be viewed as if it were a document with its "bag of words". For example, when connections are bucketed by destination domain name, the internet session is no longer a sequence of connections, but a vector of counts for each destination domain name. Bucketing transforms the textual document into a numerical representation and allows machine learning to be applied.

Within the text mining domain, the active learning method *Clustering By Intent* (CBI) [25] has been developed to overcome the problems of semi-supervised clustering. With some clusters available a-priori, the algorithm aims to find new clusters by looking at documents that are very different from the existing ones. For these different documents, it searches for a word that distinguishes (some of) them from the existing clusters. While this method works well for text - where a single feature (word) can mostly describe a topic - it is likely not working for network traffic. As network traffic has more synonym features, a single feature might only capture a subset of the sessions related to an activity. On the other hand, since the containment of information is divided over multiple features, and very few features are relevant by themselves, any feature might become relevant. While a single word might be able to capture a certain topic, a single feature in network traffic is not. Therefore, the first subquestion is:

SQ1: How to find a set of features that sensibly distinguishes a group of internet sessions?

Since the model aims to learn from the feedback given by the domain experts, it is important that this feedback is used in a meaningful way. While CBI was designed with this idea in mind, it needs to be adapted to network traffic instead of textual input. Subquestion 2 is used to solve this problem:

SQ2: How to iteratively use these feature sets to cluster new groups of data points incorporating guidance from the expert?

While CBI focuses only on the algorithm for finding clusters, because documents could simply be read by the expert, this would not be appropriate for this domain. Sessions are complex data structures that cannot be understood directly. If the expert's feedback is critical for clustering, the expert should be provided with clear information to make decisions. Since sessions are not self-explanatory, it would not be sufficient to simply show them. Therefore, the cluster should be presented (visually) in such a way that it is clear why the sessions are grouped together. The visualization is treated as the last subquestion:

> *SQ3: How to visually present a cluster to the analyst such that it can be evaluated?*

1.2 Contributions

The following contributions were made by this thesis project:

1. This project shows that semantically close features (e.g. domains), in network traffic, can be determined through feature embeddings.
2. This thesis proposes an extension of CBI, adapting it to network traffic. This extension mainly involves the use of feature embeddings that allow synonym features to be considered.
3. A real world implementation of the technique on network traffic of the Dutch National Police.
4. A novel way to cluster end-to-end network traffic from several entities.

Chapter 2

Background & related work

Before looking at specific data mining techniques that help in clustering the entities in the data, the networking domain is explored first. A background in network traffic analysis is a necessary requirement for understanding design choices later. It will also show what techniques already have been applied on the network traffic. The section about the data mining methods will further shape the direction of the thesis and show why it was chosen.

2.1 Network traffic analysis

This section commences by offering fundamental background information pertaining to the nature of network traffic and an elucidation of the underlying workings of the internet. Subsequently, existing forensic methodologies and specialized tools designed for the analysis of network traffic will be discussed. Finally, an overview of machine learning applications currently deployed in the realm of network traffic analysis is presented.

2.1.1 Network traffic

When people first started using a computer, information was inputted into and subsequently retrieved from it. However, soon after, the demand for communication between computers started to exist, and *computer networks* came into existence. Computer networks are the collection of computers interconnected by a single technology, the most famous example being the internet [62]. Given that technology is created by many manufacturers, a demand emerged for a structured and uniform means of communication. As advocated for by [21], there should be a hierarchical structure of processes in order to validate larger software projects. Eventually, this has led to a design philosophy for network protocols with multiple *layers* [23]. Within this architectural framework, layer n is entrusted with a well-defined task, accompanied by the expectation of a particular output. This allows layer $n + 1$ to operate seamlessly without the necessity to concern itself with the intricacies of lower-level activities. The most well-known model that is used to reason about protocol layering, is the Open Systems Interconnection (OSI) model [74]. While the protocols of the model are not actually used any more, the model itself is very general and hence still valid [62]. The OSI model contains the following layers [62]:

7. **Application Layer.** Specifies the actual application that the user has chosen. A popular protocol is the *Hypertext Transfer Protocol Secure* (HTTPS) that is used for loading websites securely.
6. **Presentation Layer.** This layer is concerned with the syntax and semantics of the data. It makes it possible to communicate for computers with different internal structures.
5. **Session Layer.** Allows users to establish sessions between each other.
4. **Transport layer.** Accepts data from the layers above, breaks it into smaller chunks and passes them on to the network layer to coordinate the arrival at the destination. Popular examples are the *Transmission Control Protocol* (TCP) which ensures reliable data transmission, and the *User Datagram Protocol* (UDP) that is a simpler protocol without error recovery or guaranteed delivery.
3. **Network layer.** This layer finds the optimal way from source to destination. Routing packets can be highly dynamic and each of them can travel using a distinct route to avoid congestion or failed services.
2. **Link layer.** Makes sure that there are no transmission errors in the data sent by partitioning the input into smaller data frames and patiently awaiting acknowledgments from the recipient for each frame.
1. **Physical layer.** Concerned with transportation of the actual bits over the communication channel. Deals with the physical medium the traffic is routed over.

Unlike the OSI model, the *TCP/IP* model [18, 13], is a very concrete and practical model that relies on standardized protocols [23]. The model - that is named after its primary protocols *Transmission Control Protocol* (TCP) and *internet Protocol* (IP) - is the suite of protocols that are used on the internet. Instead of including all the layers of the OSI model, it includes only 4 as can be seen in Figure 2.1. In order to communicate over the internet, information is sent using messages. A message is routed over possibly multiple intermediate machines to its final destination [62]. In order for the intermediate and destination machine to understand how - i.e., with what protocols - the packet must be processed, header information is added. On the source machine, commencing from the Application layer (i.e., layer N), a layer N *protocol data unit* (PDU) is generated, comprising both the data intended for transmission and its associated protocol information. Following this, the second-highest layer (i.e., layer $N-1$), retrieves the PDU from layer N, treating it as uninterpreted data, and subsequently envelops it with its own protocol-specific information. Hence, resulting in the layer $N-1$ PDU. This iterative procedure continues until the PDU reaches the lowest level within the TCP/IP stack [23]. This is further illustrated in Figure 2.2. At each receiver, the data is processed in the opposite direction: the layers are decapsulated such that the information can be used. Most of the intermediate machines only need to read the lower level layer information for routing the message, the receiver, however, will decapsulate the full unit. On the network layer (OSI layer 3), a PDU is called a (network) “packet”.

Given that this is the most common form for internet communication reference, this will be the common form used during the thesis. However, specific naming for messages on different layers do also exist. A PDU is called a “frame” on the data link layer (OSI layer 2) and “segment” on the transport layer (OSI layer 4) [60]. How insights can actually be retrieved from network traffic flows will be discussed next.

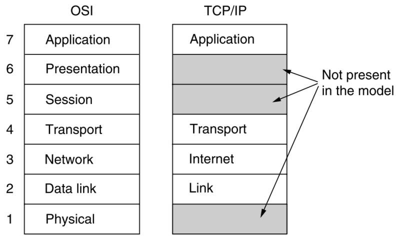


Figure 2.1: The TCP/IP model in regard to the OSI model [62].

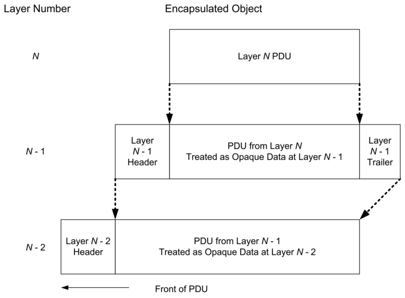


Figure 2.2: Illustration of encapsulating data from higher layers in lower layers. [23].

2.1.2 Network forensics

Given the significant role that the internet plays in the present time, a lot of important information is sent over it. Criminal offenders are also twofold dependent on the internet. On one side, criminal activity is more and more conducted within the digital domain. Examples of these are Ransomware attacks [50] and online child abuse [15]. However, for offenders of conventional types of crime the internet has become crucial for their activities as well [56]. For that reason, intercepting internet traffic of the activities from the criminals can be very useful in ongoing investigations performed by Law Enforcement.

Applications and even the operating system can be tricked by the signal they receive if malware is effective. The signal itself transmitted over the wire, however, remains veracious, as nothing can affect it [46]. That is why *packet capture* - the activity of intercepting network packets - is so effective. Network packets can be captured by *packet analysers*, software tools that intercept and log the traffic traversing over the network. These software tools can have access to the network packets by putting the capable wired network interface controller (NIC) into promiscuous mode. This means that all traffic routed over the controller can be directed to the CPU rather than the frames it was programmed to receive. In order to store the captured traffic, the packets can be written to a *libpcap* (pcap) file. While different implementations exist, all pcap files have a similar structure. A global header includes, among others, information about the time zone, data link type and timestamp precision. Then, a - by timestamp - ordered list of n packets is included as follows: $[(\text{packet header}, \text{packet}) \mid i = 1, 2, \ldots, n]$. In here, the *packet header* contains, among others, the timestamp and size of the packet [60].

In order to go from the low level packet representation to something more interpretable, several tools have been developed. The tools are not specifically developed for activities deployed by Law Enforcement. Instead, the majority of the literature in this domain is oriented towards private sector interests. Objectives of these endeavours encompass activities such as malware detection [58, 72] and incident response [55]. Given the multitude of available packet analysers, discussion of them all would be infeasible. For those readers who wish to explore this subject further, a comprehensive list can be found in [60]. Certain select analysers, deemed of particular significance, will be highlighted, however.

- **Wireshark**[1], formerly referred to as Ethereal, is the most popular open source packet analyser. It can either be used for real-time analysis (by capturing data from the network) or offline analysis (by reading pcap files). Using its graphical user interface (GUI), network analysts can see all incoming traffic in different colours. These colours, among others, can be used as warnings to highlight certain packets or help identify the particular protocol of the packet quicker. By using filters, an analyst can look for specific packets that are deemed necessary for the particular investigation. Wireshark supports more than 750 distinct protocols, with a consistent influx of new protocols being incorporated, driven by the active contributions from the community [9]. It is frequently commended for this user-friendly and intuitively designed GUI [28, 6]. Nonetheless, it lacks the capability to alert the user when anomalous network behaviour is detected or to manipulate the network traffic [6].

- **Tcpdump**[2], is a command line tool that is built on top of libpcap. Using various options, users can retrieve information from the network traffic data in different formats. While it works very efficient and fast, it is not as user-friendly as Wireshark [28, 7].

- **Zeek**[3], formerly known as *Bro*, is a platform and network analysis framework. It captures, dissects, and interprets network packets, providing detailed insights into network protocols, connections, and the data being transferred. The technique also provides its own scripting language that enable users to tailor the detection or analysis capabilities to specific needs. It is split in two main components. The "event engine" transforms network traffic into sequences of higher level events. The "policy script interpreter" handles the events based on policies created with the dedicated scripting language [54].

The mentioned network analysis tools can be helpful for digital investigators in two ways. Firstly, they facilitate the examination of network traffic directed towards a specific cybercrime victim, enabling an inquiry into the methodology employed in the execution of the criminal act and the identification of the responsible entity. The second use-case for Law Enforcement to make use of these tools is to monitor intercepted internet traffic of a particular suspect. Monitoring the internet traffic could help to identify the natural person, or to find evidence of the committed crime. Due to the growing prevalence of encrypted network traffic, this has become a very challenging activity [3, 65]. Unlike the interception of telephone traffic, the investigator cannot record the content of

[1] https://www.wireshark.org/
[2] https://www.tcpdump.org/
[3] https://zeek.org/

the communication. Instead, analysing the metadata of the internet traffic (i.e., the information contained in the headers) is an important source of information [3]. However, analysing this traffic with the conventional tools is hardly impossible for investigators that are no expert in digital forensics [65]. The authors proposed a new technique that is easier to use for non-experts. Moreover, the existing tools under consideration are primarily oriented towards conducting comprehensive individual investigations. The dataset of this thesis encompasses information pertaining to a substantial number of entities, rendering it impractical for individualized research efforts at this scale. Consequently, there is a need for a tool capable of identifying noteworthy patterns across a broader spectrum of entities.

2.1.3 Machine learning approaches for analysing network traffic

Considering that network traffic consistently entails a substantial volume, denoted as "the quantity problem" [16], the manual inspection of all data is rendered impractical. Previously mentioned tools solved this by filtering specific interesting packets of the traffic based on characteristics. However, this approach can potentially result in a significant amount of overlooked information due to a lack of a priori knowledge regarding the discernment of the pertinent segments of the data. For this reason, as in many other fields, machine learning methods are being developed to help to analyse the data. In this section, two important fields will be discussed.

A prominent application of machine learning within the domain of network traffic analysis is observed in the context of network intrusion detection systems (NIDS). While these systems used to work with complex rule based systems, machine learning is now required to detect the more complex attacks [38]. Changing feature distributions in train and test data is a major problem for models trained in this field. Once malicious traffic has been detected and attackers change their behaviour, models do not work as well any more without being retrained. As malicious traffic is subject to continuous variety, this is a major bottleneck [39]. For that reason, researchers have studied models that can adapt to new situations; i.e., that can transfer to a new domain [39, 8, 72]. An additional challenge is the substantial dimensionality of the data, necessitating the implementation of dynamic and computationally efficient dimensionality reduction techniques [52].

Another important machine learning research area is traffic classification. Traffic classification refers to the activity in which a program aims to predict the source (i.e., application or protocol) the traffic is produced by [2]. It can be used for a variety of applications, including pricing in internet service providers, anomaly detection and Law Enforcement investigations. Three categories can mainly be distinguished [2], i.e., (1) port-based classification methods that simply look at the port registered to, (2) payload-based classification methods that look at the (encrypted) content and (3) flow-based classification methods that look at unique time-related patterns of the internet traffic. While deep learning methods have the advantage that no features need to be engineered, the need of a large amount of labelled data makes them hard to use. An example of traffic classification is the determination of VPN vs non-VPN traffic. This helps enterprises making sure that connections are only established from non-VPN machines [49]. If individuals seeking to compromise confidential information were to establish a connection through a Virtual Private Network (VPN), the task of tracing their activities becomes considerably more challenging. Another form of traffic classification, DDoS attack classification, helps websites in protecting against the undesired traffic [68].

While some of the techniques - e.g. traffic classification - can be useful for Law Enforcement as well, none of them are concerned with the problem at hand. No papers could be found that analysed, compared or clustered a larger group of entities end-to-end; i.e., from source to destination.

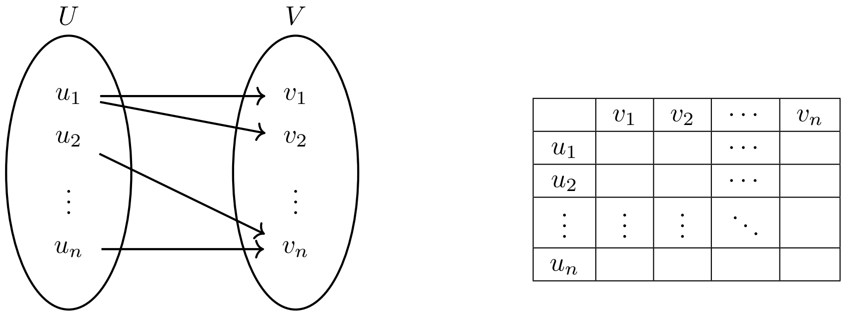
2.2 Data mining techniques

In this section, several potential data mining techniques are explored. Some of these techniques are discussed due to their important role in the methodology of this thesis, while others are examined to illustrate their inapplicability to the specific context, thus elucidating the rationale behind the chosen approach.

2.2.1 Bi-clustering & bipartite graphs

Normal clustering is only applied to one mode of the data; i.e., the features. Bi-clustering - also referred to as co-clustering - on the other hand, is a technique that utilizes both modes (i.e., the objects and the features) of the data to find relevant partitions [27]. It was initially employed in the analysis of gene expression data to identify co-expressed genes and continues to be predominantly utilized in the realm of biological and biomedical research [69]. A big advantage of bi-clustering is that it is able to find clusters of objects based on a subset of features instead of all features. While co-clustering works with any type of tabular data, i.e., objects with any kind of features, it is often applied in situations where features are some kind of independent objects as well [27]. Typical examples are expression level of genes (rows) and experimental conditions (columns) in bioinformatics, documents and bag-of-word representations in text mining, and users and pages in web mining. The data at hand - the network traffic - could potentially also be abstracted as such: entities and their target destinations.

The latter also provides an opportunity for a particular representation of the data: bipartite graphs. A bipartite graph $G = (U, V, E)$ is a type of graph where with nodes are either in U ($u \in U$) or in V ($v \in V$). Edges ($(u, v) \in E$) can only be between nodes of different domains. Unlike conventional (homogeneous) graphs, edges between nodes of the same domain do not exist (e.g., $(u, u) \notin E$). An illustration of a bipartite graph be seen in Figure 2.3a. The graph can be regarded to as a matrix with rows U and columns V [70]. This yields the adjacency matrix that shows the strength of the interconnections between nodes required for co-clustering. An example can be seen in Fig. 2.3b. While many studies employing bipartite graphs focus on statistical properties, in [43] a framework was proposed that takes the local interactions between the nodes into account. A high level process of Problem Characterization, Model Construction and Model Analysis & Evaluation is presented. This process guides researchers



	v_1	v_2	$\cdots$	v_n
u_1			$\cdots$	
u_2			$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
u_n				

(a) A bipartite graph

(b) Adjacency Matrix for bipartite graph

Figure 2.3: A bipartite graph and its corresponding Adjacency matrix

to an eventual visualization of the data while there is room for several techniques at every step depending on the domain.

A problem however, is that the network traffic has more than a single class of features that would be important for determining the entities' goal; i.e., a connection is not just defined by its destination. For that reason, it would be important to enable multiple features. One study tried to extend the model to accept tripartite data. In [42], COVID tweets were clustered by first clustering the user-topic bipartite graph. After that, tweets were assigned one of the user clusters based on majority vote. In [73], both the inter-level (comparison of clusters based on two node-types; bipartite graphs) and intra-level (comparison of clusters from both bipartite graphs) were taken into account. A deficiency of co-clustering on bipartite graphs is that it can only learn from the connections between nodes; information *about* nodes is lost. In [35], node attributes were included in the clustering process to incorporate the information about the nodes themselves. For the network data, however, most information is stored in the actual connections; i.e., in the edges. While bi-clustering can leverage information regarding the interconnections between nodes and may even have the capability to extend the nodes themselves, it does not account for information pertaining to the actual edges. For that reason, this technique is not suitable for the network traffic application and a more general technique is required.

2.2.2 Clustering high-dimensional data

In high-dimensional data, there are many features (ranging from several thousands to several millions) associated to every observation. This comes along with new challenges, and hence requires specialized techniques compared to conventional methods [57]. These challenges are caused by *the curse of dimensionality*. Among others, the computational time can increase exponentially [51], and neighbours can become meaningless when the data increases [11]. Considering the high-dimensional nature of the dataset presented in this thesis, an exploration of these techniques becomes a natural choice. Two main approaches can be distinguished [51]:

Dimensionality reduction. By reducing the number of dimensions, the data can be converted to a lower dimensional space. By reducing the data to 2 or 3 dimensions, the data can be visualized on a screen. It is common practise to use dimensions reduction for the first - explorative - stage of data analysis [22]. Multiple methods are available, with notable models including Uniform Manifold Approximation and Projection (UMAP) [45], Principal Component Analysis (PCA) [44], and t-distributed stochastic neighbour embedding (t-SNE) [66]. While the method offers a convenient means of early-stage data exploration, it is worth noting that [19] has identified a significant potential risk. Their study showed that, after extreme dimension reduction from hundreds or thousands of dimensions to 2 or 3, significant distortion takes place and results are not reliable.

Clustering methods. A major branch within the clustering methods for high-dimensional data is subspace clustering. As there are many dimensions, different clusters are not necessarily based on the same set of features. Where some clusters exhibit cohesion based on some specific features, distinct data points form a cluster based on differing features. Subspace clustering, hence, extends normal clustering by searching clusters in different subspaces [53]. Two main types of methods exist. Bottom-up subspace search methods start with individual data points and gradually combine subgroups to find higher level patterns in the data. A noticeable example of such method is CLIQUE [5]. The second category of subspace clustering methods, iterative top-down subspace search, has a reversed approach. These methods, such as PROCLUS [4], start with equally weighted dimensions and approximate an initial clustering using all dimensions. Next, an iterative process is initiated in which feature weights are updated such that clusters are regenerated [53]. Another category of clustering methods is correlation based methods [51]. These methods combine dimensionality reduction algorithms with clustering approaches. They focus on the relation between objects rather than the feature representation of the objects. These methods can be categorized in the dimensionality reduction method used [36]: PCA based or Hough transform based. The previously discussed bi-clustering methods are also considered a category of high-dimensional clustering [53].

The discussed high-dimensional clustering methods are not suitable for the problem at hand. Dimensionality reduction could at most serve as a preliminary step in the data analysis activity. The actual clustering methods, suffer from overlapping clusters, caused by noisy environments [53]. Given the substantial presence of noise in the network traffic, it is important to have control over the clustering process. The control is usually not provided by the high-dimensional clustering methods. For that reason, methods that do provide control and input from domain experts are to be discussed next.

2.2.3 Semi-supervised clustering

Traditionally, clustering was an unsupervised learning method. As domain experts could often enrich the clustering by adding their prior-knowledge, the field of semi-supervised clustering (SSC) emerged [14]. Within this field, there are two main categories. (1) SSC algorithms that have access to some labelled data, but too little to do proper classification. In this case, the labelled data is combined with model assumptions learned on the data to come to meaningful clusters [34, 32]. The (2) SSC algorithms that ask users for constraints on whether objects should (must-link constraint) or should not (cannot-link constraints) be in the same cluster [67, 71].

While these algorithms are able to add external information, they do have some deficiencies. In the first case, it is assumed that all clusters are already known. However, in real world situations, it might be the case that clusters are not yet included in the training data. In the second case, except for large constraint matrices having a major impact on the execution time [14], it might also not always be as simple to decide on the belonging of objects.

2.2.4 (Inter)active clustering

While normal machine learning tasks often assume sufficient (labelled) training data, active learning is a technique that is aimed to minimize the number of to-be-labelled data points [63]. Using a *query strategy*, the active learner presents data points to the Oracle (i.e., domain expert) that are most likely to be useful for the learning process. While most attention is paid to classification tasks, some papers are mentioned to be devoted to clustering as well [37]. In these papers, to-be-labelled data points are selected that could determine the purity of a created cluster. These methods however, require the user to know what classes exist in the data.

Another form of active clustering is a variant of semi-supervised clustering. The application created in [70], shows clustering results to the user and accepts cannot- or must-links from the user. Hence, the application supports an iterative learning process based on human feedback. A strong focus of the paper is on the visualization of the results. By showing heat- and treemaps as well as attribute statistics of the clustered items, the user should be enabled with enough information to make the desired decisions. The visualization of the data worked well for the particular application; co-clustering senators and bills. However, for less self-explanatory data, such as internet data, these visualizations would not be satisfactory.

To overcome the problem of getting undesired clusters, *Clustering By Intent* aims to cluster the data using the intent of the user [25]. The method is able to find new relevant classes based on a labelled training set which means that not all classes must be known a priori. Every time that the learner found a new cluster, the oracle can choose to accept, merge or decline the cluster. CBI starts with documents that should be clustered based on the topic. Each document $x \in D$ is represented by its bag of words; i.e., each word is a feature f in feature space (F). For some documents, a label is available. All labelled documents together are the training set $T \subseteq D$. Each labelled document is assigned a topic, a cluster $c \in C$. The rest of the data - the unlabelled data - is referred to as $U \subseteq D$ such that $U \cup T = D$. For every document x, the best two predictions are compared. If the difference between these two predictions is small, they are considered the residual set R. For this residual set, the goal is to find a feature, f that distinguishes a group of documents $R_f \subseteq R$ from T with the highest divergence. As long as $|R_q|$ is larger than a set threshold, a new feature $f' \in F$ is added such that only documents that also contain f' remain. For every $f \in F$, divergence is calculated by taking the precision ($\frac{|R_f|}{|R_f|+|T_f|}$), from documents with feature f in R (i.e., R_f) compared to documents with feature

f in T (i.e., T_f).

While this positive feedback is used (i.e., clustered data is added to the training data), negative feedback is ignored. In order to utilize this feedback, [10] improved the technique by collecting all features that constituted to declined clusters. Not only are these features deferred from future clusters, similar features are also considered weaker when determining new clusters.

2.2.5 Feature embedding

Clustering By Intent works under the assumption that a single word (feature) can sensible distinguish a group of data points from different data [25]. When the data mainly consists of features that occur very little, a single feature could only distinguish a very small group of entities. From this point, no difference can be made between noisy (irrelevant) features and features that are relevant. In this situation, it would be important to find “synonym” features, features with a similar meaning. While a single feature might not be able to distinguish a group of entities of sufficient size, a set of synonym features might.

Research in comparing features has mostly been focussed on text. *Word2vec* is a technique that learns the meaning of a word by looking at the context it is written in [47, 48]. Two different architectures where initially proposed by the authors. The first architecture, *Continuous Bag-of-Words Model* (CBOW), looks at nearby words disregarding the exact position. A Feed-Forward Neural Network was trained using the training criteria of predicting a word given the 4 words before and after. The second architecture that they introduced is the *Continuous Skip-gram Model* (skip-gram). This model, given some word, aims to maximize classification of the surrounding words. Unlike CBOW, this Feed-Forward Neural Network does look at word positions. As a word's meaning is stronger determined by the words close by, these are given a stronger weight than words more distant. On both sides, a randomly decided 1-10 words were predicted for every current word. Besides showing good results in a sentence completion challenge, they also showed captured meaning of words differently. They showed that they could apply vector arithmetic on the word embeddings yielding sensible results. For example, $\texttt{Paris} - \texttt{France} + \texttt{Italy}$ results in `Rome`.

The underlying technique behind *Word2vec* has been extended to many more applications. The authors themselves used it to learn representations of paragraphs [40]. The technique, however, can also be used for non-textual purposes, and has demonstrated itself as a robust technique in many ways. In *prod2vec* [29], it has been used to determine product recommendations based on incoming purchase emails. Other applications include *node2vec* [31] for continuous feature representations for nodes and *Tile2Vec* [33] for spatial distributed data.

Interestingly, *Word2vec* has also been extended to the internet traffic domain. In [24], researcher used it to detect malicious traffic. Firstly, segments from the Transport Layer Security (TLS) protocol were transformed into words, comprising a fusion of multiple attributes. Each word was then converted into an embedding by training Word2vec on the transformed TLS sequences. Using a recurrent Neural Network (RNN), they predicted whether the TLS sequences - now consisting of frame embeddings - where malicious or benign. Another study used the (textual) payload from HTTP traffic for anomaly detection [41].

They trained word2vec on the HTTP payload documents first. Then, for every payload, they obtained the document embedding by taking the weighted TF-IDF average of its words. Subsequently, the "payload embeddings" were used as feature space in the anomaly detection model.

2.2.6 Topic modelling

Topic modelling is a set of text mining techniques that is about finding underlying themes (i.e., topics) in a set of documents [1]. It regards to documents as bag of words; a collection of words where order is ignored while frequencies are accounted for. The most wellknown topic modelling method is Latent Dirichlet Allocation (LDA) [12]. In this method each document is seen as a mixture of topics and each topic is seen as a mixture of words. Hence, from the document-word matrix, a document-topic matrix and a topic-word matrix are created using the Dirichlet distribution. LDA uses an iterative process in which these two matrixes are optimized such that the probability that these two matrixes yield the original document-word matrix is maximized. Embedded Topic Model (ETM) was developed to extend LDA by using word embeddings [20]. Instead of working with a topic-word distribution matrix, ETM initializes topic embeddings that are to be optimized by comparing them with Word2vec embeddings that are trained beforehand on the vocabulary. Using the embeddings, topic modelling should be more resilient against large vocabulary sizes [20]. Challenges for topic modelling are visualizing and interpreting the models [30]. LDAvis [59] is a tool that addresses this problem by providing a user interface in which topics and their most relevant words can be explored.

Chapter 3

Embedded Clustering by Intent

This section begins with a comprehensive description of the data model, while providing formal notation to ensure clarity for the reader. It then presents the problem statement before discussing the implementation of ECBI.

3.1 Data Model & Notation

This section explains the data model and necessary notation used throughout the thesis document. Since this thesis is not concerned with the low-level representation of network traffic, additional information on processing to obtain the higher-level data model is also provided.

The raw network traffic data is retrieved from packet capture files and then processed with Zeek[1]. The result of this process is a set of network packets that are aggregated into a higher-level network event with information about the higher-level headers and group statistics of the lower-level packets. These events are always initiated by an entity:

Definition 3.1. An entity $e \in E$ refers to an identifiable agent in the data. In this context, "identifiable" does not necessarily refer to the identification of a natural person. Rather, it simply means that Internet traffic from e_1 can be distinguished from traffic initiated by e_2.

These higher-level network events can now be formally defined as "connections":

Definition 3.2. A connection is the most basic component within the data. It is a consolidated event created from a sequence of network packets generated by Zeek. This higher level event incorporates information from the upper level headers and aggregates statistical information from the lower level packets. It is created by a single entity and always has a destination. Formally, for a connection c, $\neg(\text{EST}(e_i, x) \wedge \text{EST}(e_j, x)) \mid e_i, e_j \in E, i \neq j$. Here $\text{EST}(e, x)$

[1] The background on the general functioning of this process can be found in Section 2.1.

means that c is established by the entity e. An illustration of a connection can be seen in Figure 3.1.

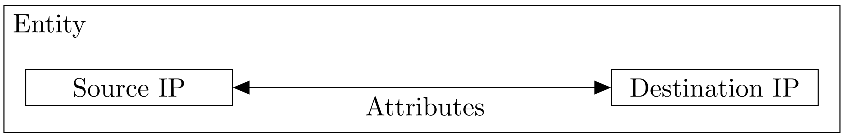


Figure 3.1: Illustration of a connection

High-level header information includes source and destination IP addresses, application protocol, and port number. The aggregate statistical information includes the aggregate incoming and outgoing byte sizes, start and end time of the packet sequence. In addition to the above information, other attributes such as geolocations retrieved from the IP address - i.e., enrichments - are provided. A detailed discussion of how to obtain these attributes is beyond the scope of this thesis. However, since the 'domain' field will be used extensively and its assignment is not immediately obvious, it needs further elaboration.

There are 3 methods to populate the domain field for each connection. The methods are presented in order of priority, i.e. method 2 is only used if method 1 does not return the domain field.

1. The first method should retrieve the domain field from the SSL certificate present in the Zeek event.
2. For `HTTP` traffic, the domain can be obtained from the `hostname` application layer header sent with it.
3. The main source of information for retrieving the domain is the use of Domain Name Server (DNS) traffic, which is found throughout the captured network traffic. Domain Name Servers allow an Internet user to type a URL into a browser and have it translated into the corresponding IP address. By caching these domain-IP pairs from the DNS traffic, the domain can be looked up in this cache for each new connection encountered. With this method, the domain field is always set to the most recent lookup value of an IP address.

All of these attributes reveal something about the intent of the entity initiating the particular connection. In this thesis project, attributes are not used as such. Rather, features, as defined below, will be used throughout the project:

Definition 3.3. A feature $f \in F$ describes a connection so that it can be compared to other connections. Therefore, it works to group a set of connections. For example, if only domains are considered, then $f_{google.com}$ denotes the number of connections to the domain `google.com` made by e. F can also be thought of as the abstract version of a feature space in a bag of words manipulation of textual data.

Internet connections are not solely established, but are located in a sequence. In this dissertation project, such a sequence is called an Internet session:

Definition 3.4. A session - short for internet session - $s \in D$ is "a finite uninterrupted amount of time in which a single entity uses the internet to access, share, and exchange information and services through Internet-enabled devices". More formally, $s \in D = [\ x_0,\ \ldots,\ x_n \mid 1 \leq n \leq \infty\ ]$ and $\neg(\mathrm{EST}(e_i, s) \wedge \mathrm{EST}(e_j, s)) \mid e_i, e_j \in E, i \neq j, s \in D$ where $\mathrm{EST}(e, s)$ denotes that session s belongs to entity e. The session is regarded uninterrupted if the delta of starting times of two successive connections never exceed a threshold referred to as `session gap time`. If two consecutive connections are described by the same feature, the latter is removed.

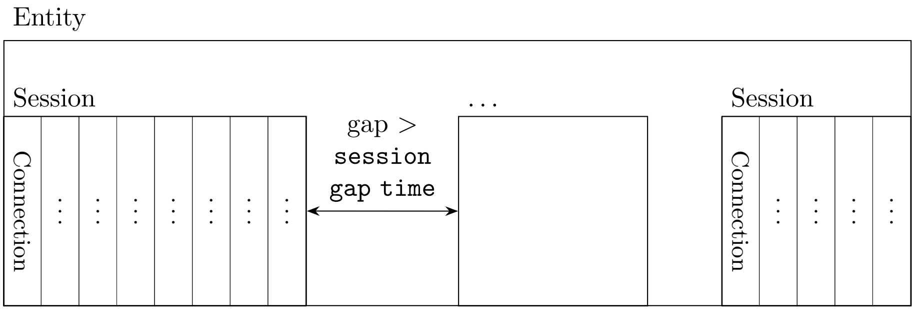


Figure 3.2: High level overview of the data

Figure 3.2 further illustrates the relationship between e, s, and x. A group of sessions that share a common feature is defined as follows

Definition 3.5. $D_x^y \subseteq D$ is the set of sessions for which feature $f_x \geq y$. However, the abstraction D_x can be used to refer to the set of distinguished sessions as if f_x were a boolean value.

The last concept to be introduced is the query q:

Definition 3.6. A query q consists of a set of features $f_x \in F$ and indicative values. The value y indicates the minimum value for a feature to be true. The values n and m determine the size of F_q (i.e., $|F_q| = n$) of which m must result in true. A session s is selected by the query if the following is true: $\sum_{x \in F_q} 1(s \in D_x^y) > m$. A set of sessions distinguished by q is denoted as D_q.

3.2 Problem statement

The goal of this thesis project is to find clusters of sessions where for each cluster $c \in C$ an activity (as opposed to e.g. the operating system of the entity) should be central. If the activity for some session $s \in D$ is similar to that of c, it should be included. If the activity is different, it should not be included. Therefore, the probability that s is included in c must increase as the activity of s and c becomes more similar.

More formally, for each $c \in C$ and each $s \in D$ the following is to be obtained:

$$\lim_{\mathrm{d}(a_s, a_c) \to 0} p(s, c) \to 1$$

In the above, a_x denotes the activity of x, $\mathrm{d}(x, y)$ the difference between x and y while $p(s, c)$ indicates the probability that s is in c.

3.3 Methods

In this section, the methodology of the thesis project is discussed. A high-level overview of ECBI is presented first, after which each sub-question is addressed in more detail.

3.3.1 Overview of the method

The goal of this thesis project is to sensibly cluster entities based on their features. This is done by using an iterative process based on the Clustering by Intent [25] method discussed earlier. The method, a simplified illustration of which is shown in Figure 3.3, includes the following steps

1. At the beginning, the analyst creates at least two clusters $c \in C$ of Internet sessions in the data set D. All sessions s that are part of a cluster are considered to be in the training set $T \subseteq D$. For these sessions, it is assumed that the analyst knows what the entity's goal was. In Figure 3.3a, a session s is represented by a circle, a color indicates that $s \in T$, and each color represents a separate cluster. Furthermore, the table illustrates that each session is characterized by its features, i.e., its 'bag of connections'.

2. In the second step, a Multinomial Naive Bayes (MNB) prediction is performed to find the set of sessions that are least likely to belong to an existing cluster; i.e., the residual set $R \subseteq D$. In Figure 3.3b, R is represented by all circles that are still black.

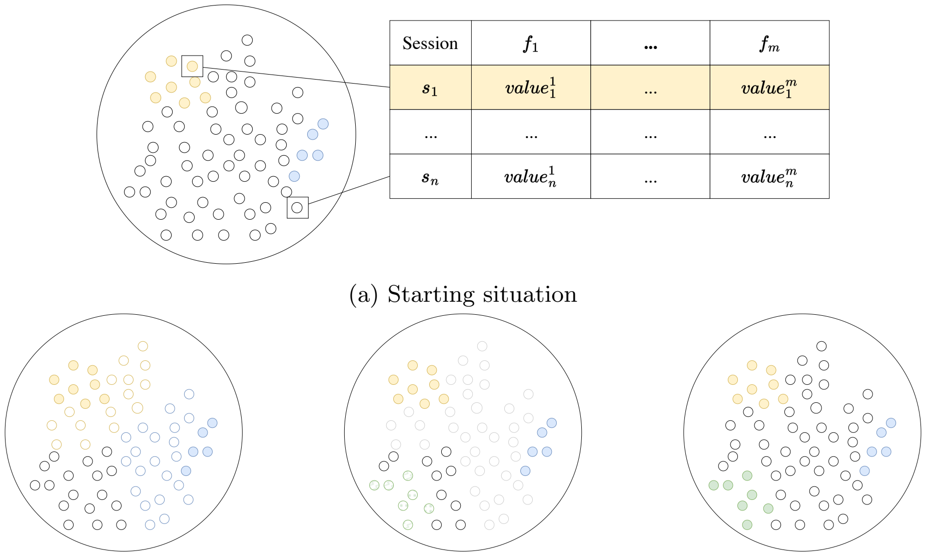
3. The third step is to find a set of similar features (i.e., the query q) that could be the prerequisite for a new cluster. Using q, a set of sessions $R_q \subseteq R$ is selected. The same query is run on T to see how different R_q is from T. In Figure 3.3c, R_q is represented by the green color and the fill patterns represent different features $f \in q$.

4. The combination of q and R_q that yields the most distinct cluster is presented to the analyst. The analyst can accept the cluster, reject it, or merge it with an existing cluster. Accepting the cluster is illustrated by the situation in Figure 3.3d. Rejecting the cluster removes all $f \in q$ from F.

In the next sections, research question 1 focusses on how to distinguish a cluster of sessions, which is step (3) above. Research question 2 discusses the overarching ability to learn from the feedback provided by the analyst; it discusses the entire process shown above. The third research question provides more detail on how the analyst is empowered to make the decision for step 4.

3.3.2 Research question 1

Due to the sparse nature of the data, many features do not occur often. More formally, for many $f_x \in F$, $|S_x|$ is small. On the one hand, many features that might be interesting for clustering cannot be used because they do not occur often enough. On the other hand, if small clusters were accepted, every f_x could become relevant. Only features with a certain degree of robustness need to be selected. Robustness can be achieved if the selected feature discriminates



Session	f_1	...	f_m
s_1	$value_1^1$	...	$value_1^m$
...	...	...	...
s_n	$value_n^1$	...	$value_n^m$

(a) Starting situation

(b) Situation after MNB classification

(c) Finding new cluster in the residual set

(d) New situation if cluster is accepted.

Figure 3.3: High level overview of the proposed method based on Clustering by Intent.

a sufficiently large set of entities. There always is a feature that perfectly discriminates some sessions. However, this may well be a noisy feature that should be ignored.

This is different from the situation in [25], where examples are given where a single word could separate documents from the others by a single feature. The main difference in purpose is that CBI should find clusters based on topics within a single topic. In this case, there are features that have sufficient occurrence to serve as a rational discriminator by themselves. This is further illustrated by Example 3.1.

Example 3.1: Finding Topics

. The goal is to cluster sports text based on the actual sport; for example, clustering sports news about football, hockey, etc. In this situation, using the word "Football" would be sufficient to find all related documents, since the sport is usually mentioned in the article. However, clustering the news articles based on the more general topics (e.g., 'sports') would be much more difficult. These general topics are often not mentioned in the texts, and therefore no single feature can be found.

So instead of using a single feature, multiple features need to be combined. This is especially important when the data contains many synonyms.

Definition 3.7. Synonym features (i.e., synonyms) are features that have a similar meaning and are therefore present in similar Internet sessions. Formally, for synonym features f_a and f_b, $S_a \sim S_b |\ S_a, S_b \subseteq S$.

Synonym features are especially important for Internet data. Many services exist that provide similar services, as can be seen in Example 3.2.

> **Example 3.2: synonym features**
> . Consider two social media domains x and y. A connection to x (i.e., c_x) and a connection to y (i.e., c_y) could identify similar activity. However, c_x and c_y are considered to be completely independent connections.

By combining several of these synonym features using an *or* relation, a larger set of sessions with similar intentions can be found:

$$S_x \cup S_y \supseteq S_x, S_y \tag{3.1}$$

By applying this operation, a set of sessions can be clustered around a topic that cannot be identified with a single feature. In other words, this operation can uncover the underlying topic behind a group of individual features.

However, no further details were provided on how synonym features are determined. Therefore, more support is needed. In [47], the meaning of a word is learned from its context.

Definition 3.8. The context of x_i is determined by a window (k) of preceding (i.e., $x_{i-1}, x_{i-2}, \ldots, x_{i-k^-}$) and successive objects (i.e., $x_{i+1}, x_{i+2}, \ldots, x_{i+k^+}$) that are assumed to contain information about x_i. In general k^- and k^+ are equal. However, given that the context is always restricted by the size of the sequence, this differs at the start and finish of a sequence. For a sequence of objects $x[0:n]$, $k^- < i$ and $k^+ + i < n$.

This concept of obtaining meaning from nearby data points can also be applied to the network traffic; *connection2vec*. As the network traffic is also observed in sequence, each connection has its context by preceding and succeeding connections. This context might exist of automated traffic to content providers or third parties that provide a service to the requested connection. The context might also exist of connections that were initiated by the entity just before or after. The former can provide value as similar websites might make use of automated traffic caused by similar services. An intuition for the latter is provided in Example 3.3.

> **Example 3.3: *Meaning* of internet connections**
> Imagine two holiday booking websites f_x and f_y. While different entities e_i and e_j never go the same website they do share a website f_z to look at reviews of venues. For that reason, f_x and f_y correlate because f_z is nearby in both cases.

In order to find a new meaningful cluster, a query q needs to be generated. While y, n and m are given by the analyst, F_q is generated in ECBI. Using the connection2vec method it is possible to find the $n-1$[2]most similar features to some feature f. Generating queries for each feature with its $n-1$ similar

[2] Note that $n-1$ similar features add up to n if the feature itself is included.

features would mean that there are $|F|$ queries to be generated. To reduce this number, F_q is only generated based on features not yet used in F_q before.

From the set of queries Q, the query that is most likely to find a new interesting cluster in R should be selected. Q is chosen in such a way that precision is optimized, adopting the choice from the CBI. While this was not specified in the CBI paper, the choice has been made to calculate the precision relative to the size of R and T. Hence, q is chosen so that $\operatorname*{argmax}_{q \in Q} \frac{|R_q|/|R|}{|R_q|/|R|+|T_q|/|T|}$. In the Clustering by Intent paper, a predetermined maximum size of 25 was used for the proposed clusters. However, it is noteworthy that instead of a maximum size, a minimum size is used in the method proposed in this thesis. This minimum size should ensure that the cluster is general enough and does not provide information about an entity specific pattern. Pseudocode for the complete procedure can be found in Algorithm 1.

Algorithm 1: Procedure to find a new cluster

```
input  : Set of features $F$, residual set $R$, training set $T$, number of
         features $n$, minimal number of features $m$, minimal value of a
         feature $y$.
output: The query $q$, and the sessions of the new cluster $R_q$
$F' \leftarrow F$;
$Q \leftarrow \{\}$;
while $|F'| > 0$ do
    $f \in F'$;
    $F_q \leftarrow f \cup$ getNMostSimilarFeatures(F, f, n-1) ;      // See Def 3.7
    $F' \leftarrow F' - F_q$;
    $Q \leftarrow Q \cup F_q$;
end
$q \leftarrow ()$;
$R_q \leftarrow \{\}$;
bestQueryScore $\leftarrow 0$;
foreach $F_q \in Q$ do
    $R'_q \leftarrow \text{SelectSessions}(R, F_q, m, y)$ ;      // See Def 3.6
    $T_q \leftarrow \text{SelectSessions}(T, F_q, m, y)$ ;      // See Def 3.6
    precision $\leftarrow \frac{|R'_q|/|R|}{|R'_q|/|R|+|T_q|/|T|}$;
    if precision > bestQueryScore then
        bestQueryScore $\leftarrow$ precision;
        $q \leftarrow (F_q, m, y)$;
        $R_q \leftarrow R'_q$;
    end
end
```

3.3.3 Research question 2

Each iteration trains a MNB model on the ‘bag of connections’ dataset. CBI transforms the data into binary values indicating whether the feature is present in a text. In this project, the data is instead min-max normalized to preserve the quantitative information inherent in the dataset.

Then, similar to CBI, a MNB is trained to find the probability that each session belongs to a pre-existing cluster. This likelihood is determined from the difference in classification score between the first and runner-up class. A small difference indicates that a session is unlikely to belong to either class. The 20% of sessions in $U = D - T$ that are least likely to belong to an existing cluster are selected as the residual set R. The method in research question 1 is used to determine q and R_q.

Both $F_q, m, y \leftarrow q$ and R_q are presented to the analyst and an interaction is expected from the analyst. The analyst has three different options for providing feedback:

1. Accepting the cluster. By accepting the cluster, C is appended with the new cluster and T is appended with R_q.
2. Rejecting the cluster. The cluster is not maintained and at least 1 of the features in F_q is deleted.
3. Merging the cluster. The sessions R_q are assigned to an existing cluster in C and T is appended with R_q.

Algorithm 2: Active network traffic clustering procedure

```
input  : Set of all network sessions $D$, set of Features $F$,
output: Set of clusters $C$, Training data $T$
$T, C \leftarrow$ askAnalystForAPrioriClusters($D$);
$U \leftarrow T - D$;
$Q \leftarrow \{\}$;
while Analyst did not finish do
    $n, m, y \leftarrow$ AskAnalystForParameters();
    MNB $\leftarrow$ trainMNBClassifier(T,U,F);
    $R \leftarrow$ findSessionsWithLowestConfidence(MNB, U) ;        // See text
    $q, R_q \leftarrow$ findNewCluster($F', R, T, n, m, y$) ;        // See Alg 1
    switch Analyst decision do
        case Cluster accepted do
            $T \leftarrow T \cup R_q$;
            $C \leftarrow C \cup (q, R_q)$;
        end
        case Cluster Rejected do
            $F_q, m, y \leftarrow q$;
            $F \leftarrow F -$ AskAnalystToRemoveFeatures($F_q$);
        end
        case Cluster Merged do
            $c \leftarrow$ AskAnalystToSelectExistingCluster();
            mergeCluster($C, c, q, R_q$);
            $T \leftarrow T \cup R_q$;
        end
    end
end
```

While the approach of CBI [25] is mostly followed here, it is noteworthy to mention the different approach in handling the 'reject' action by the analyst. Instead of ignoring the action, the analyst can remove one or more features to force the model to find another cluster. The complete pseudocode can be found in Algorithm 2.

3.3.4 Research question 3

An important part of this project, though not directly related to the active learning model, is visualization to the analyst. Unlike textual data (i.e. documents), which can be quickly skimmed to understand the author's intent, network traffic is not so easy to understand. Sessions can be long (i.e., contain many connections) and can be obfuscated by a few dominant connections that have a very high presence in the data. For this reason, some techniques should be used to abstract the data so that an analyst can rationally make the decisions necessary for the active learning process. The approach taken here is to provide the user with three different screens of information, ranging from very high-level information to a very detailed visualization of the sessions. Many of the decisions have been made in consultation with the analysts, as they know best what is needed to see if the sessions have been put together for the right reasons. The rest of this section is a detailed explanation of each screen.

Screen 1: High-Level Information. The first screen must provide the analyst with a quick overview of the cluster found. First, the screen provides some concise information about the size of the cluster, comparing it to the sizes of D, T, and R. This information can be used to relate the size of the found cluster to the bigger picture. Information about the number of entities involved in the cluster is also provided. This can help the analyst distinguish between clusters formed around patterns that originate from a single entity, and a pattern that can be observed across the entire dataset. Finally, for each feature, information is provided about its presence in R_q, T, and D. With this information, the analyst can understand how each feature contributes to the new cluster. In addition, this information also provides insight into how common a feature is. If the feature is relatively common in D, the new cluster may not reveal a unique pattern in the data and instead may reveal more about what is not yet in T.

Screen 2: Shifterator. Ultimately, the analyst needs insight into how the new cluster differs from the existing clusters. One technique that makes this possible is the Shifterator package [26]. This method was originally developed to compare texts by analyzing the specific words they use. For both texts, all key-value pairs (i.e. words with their word counts) are given to the framework. The key-value pairs are translated into bars that point either to the left or to the right. A bar pointing in one direction means that the key has a more significant presence in the text belonging to that direction. The framework provides several ways to calculate the bars. The method chosen in this thesis project is the 'Shannon Entropy Shift'. This shift not only shows the words that have the highest difference in relative presence, but also takes into account the surprise of a word. This means that words with a low overall presence are favored over words that occur frequently. An example of such a Shifterator diagram can be seen in Figure 3.4.

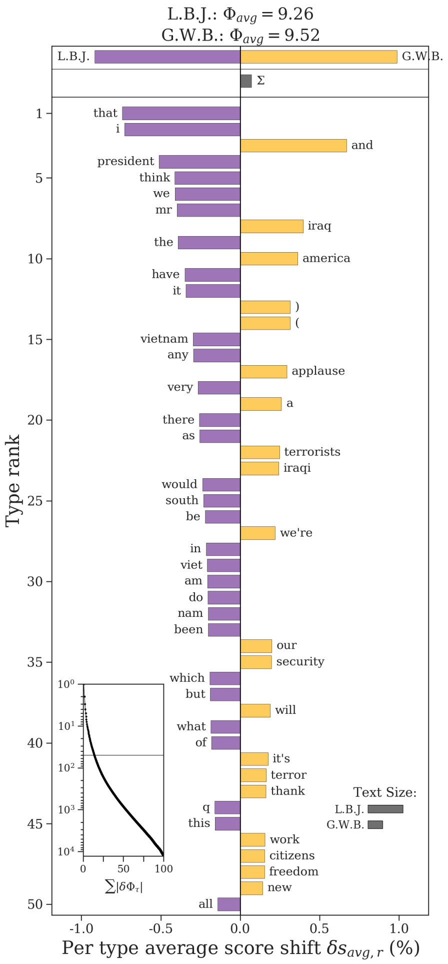


Figure 3.4: An example of the visualization, created by the Shifterator package. This example highlights the difference in words between two presidential speeches.

The necessary key-value pairs for the use of the Shifterator package are filled according to the following protocol, which facilitates the comparison between the sessions within the training set T and the newly formed cluster R_q. For both groups, the set of keys is simply set to F. For both groups, for each feature $f \in F$, the value of f is set to the number of sessions that exceed the threshold y. More formally, $f_x = \sum_{s \in T} 1(s \in S_x^y)$ for the key f in the training set T.

Screen 3: Session-level information. In addition to the first two screens, which provide group-level information, the third screen allows the analyst to test a potential hypothesis against the actual sessions. This screen is generated from the sequence of connections within a session. Each connection is represented by the attribute of the data selected to generate the features. Thus, if F were generated from just the domain field, this screen would be a sequence of domains from each session. Since the session can be very long, noteworthy connections, i.e. those that play a role in determining the cluster, are highlighted. A possible example of this screen, where the domain field is used to generate F and $F_q = \{\text{github.com}, \text{stackoverflow.com}\}$, can be seen below[3]:

> amazon.com, wikipedia.org, apple.com, cnn.com, reddit.com, nytimes.com, **stackoverflow.com**, **github.com**, ebay.com, weather.com, amazon.com, wikipedia.org, google.com, cnn.com, reddit.com, nytimes.com, **stackoverflow.com**, microsoft.com, ebay.com, weather.com

[3] Note that the sequence of domains shown is randomly generated and not from the real dataset. It is only used to show what the visualization look like.

Chapter 4

Evaluation of Connection Embeddings

The connection embeddings play an important role in identifying similar connections and hence their quality is crucial for good clustering results. This chapter contains an isolated evaluation of these embeddings to quantify the quality by comparing them against embeddings created using a conventional method. Before the actual experiment is discussed, the dataset consisting of intercepted network traffic by the Dutch National Police is introduced first.

4.1 Data

The data that is used for the evaluation is internet traffic intercepted by Dutch Law Enforcement. This data is as described in Section 3.1. While, as described there, connections have many attributes, the evaluation is only concerned with the domain field. The reason for this is that experts are most knowledgeable about the domain field compared to different attributes. Besides that, there is also an external dataset about domains that can be used to evaluate the connection embeddings, something that is not available for different attributes. For this reason, evaluating the tool would be most feasible using that attribute compared to for example the destination IP address. As discussed in Section 3.1, the domain field is not available for each connection. Each connection where the domain field cannot be populated is excluded from the evaluation set. Besides the requirement of having the domain field assigned, there can be assigned additional criteria for connections to be kept in the data. For the evaluation performed later, with human experts, it is important that the included domains could be recognized. However, the majority of domains are often only searched for by a single company, or are even typed in by mistake. To increase the probability there is an information position about the domains, only domains that are visited by at least 10 entities are contained. Table 4.1 provides insights into how the three described datasets compare in terms of their size, number of sessions, number of entities and number of domains. For each dataset, a `session gap time` of 60 minutes is used to determine the session a connection belongs to.

	Counts			
Dataset	Connections	Sessions	Entities	Domains
All	42,209,741	75,358	823	290,740
With domain field	19,567,573	72,792	747	290,740
10+ connected entities	16,605,995	64,002	741	4968

Table 4.1: Statistics about the data given different applied filters

4.2 Connection embedding experiment

In order to validate how well the Word2vec embeddings perform, an experiment has been set up. As the application is dependent on the quality of synonym connections for finding new relevant clusters, it is important to see how well it can find similar connections to a given connection. The organization Symantec provides labels[1]to domain names in order to categorize them for various purposes such as firewall implementations. Used carefully[2], the labels can serve as an external validation on how similar two websites are; i.e. whether they are categorized equally. In this experiment, by characterizing connections only in terms of the domain they were connected to, the labels can serve as ground truth in determining how well the Word2vec model proposes similar connections.

4.2.1 Setup experiment

Experiment A compares different parameters to create the Word2vec embeddings with to see how these influence the prediction quality. For each variation, the Symantec labels of the proposed connections are measured against the requested connection. If the proposed connections are mostly equal to the query connection, a high score is awarded and vice versa. The experiment is set up as it being an Information Retrieval problem [17]: how relevant are the acquired documents in response to the given search query.

The best scoring Word2vec embeddings are then tested against more naive solutions - discussed later in this section - to establish a meaningful benchmark for performance evaluation in experiment B. Within this analysis, attention is also devoted to the number of acquired connections.

The experiment, for each method, is performed as follows:

Step 1. First, a dataset where each connection has the domain field (see Table 4.1) is created. The resulting datasets contains the execution time of the connection, session ID and the domain (the only variable that characterizes the connection). All connections with a similar session ID (i.e., each session) are ordered by time.

[1] https://sitereview.bluecoat.com/

[2] Some considerations on the Symantec labels are discussed later.

Step 2. In the second step of the experiment, the embeddings are created. In case of the Word2vec embeddings, a model is trained on given parameters. For experiment A, various models are trained with different parameters.

In experiment B, word embeddings are compared against a conventional method. The conventional method involves the creation of a pivot table, where the connections serve as rows and each entity e is represented by a feature displaying the corresponding number of connections were established by e. The embedding of a connection is here the distribution that it has across all sessions. In here, the session gap time has a direct effect: it determines how many sessions (and thus features) there are in the pivot table.

Instead of using sessions as features, entities are used. While sessions would be expected given that clustering sessions is the main interest of the thesis, their large number (1M+) made it computationally infeasible to calculate.

For each set of embeddings, for all domains $d \in D$, the 10 most similar domains (i.e., $d_1^s, \ldots, d_{10}^s \in D$) according their embeddings are retrieved. Each combination of both d and d_i^s is stored. The most similar embeddings are found using cosine similarity.

Step 3. When all similar domains have been found, the labels of both the original and similar domain must be compared. This is performed using the Symantec labels. The labels give some indication of what the meaning of the website is. They have categories ranging from `Business/Economy` and `Cloud Infrastructure` to `Gambling` and `Hacking`[3]. A domain can have more than one label. The experiment is performed with the assumption that if a domain has a certain label, the obtained similar items must share at least one label. Formally, for each combination of domain d and proposed domain d_i^s:

$$|L(d) \cap L(d_i^s)| \geq 1$$

with $L(x)$ being the label(s) of domain x.

Step 4. The final step is about determining the score of the embeddings. The following metrics are used to calculate the score of a single embedding [17]:

$$\text{precision@}k = \frac{|\{k \text{ retrieved domains}\} \cap \{\text{relevant domains}\}|}{|\{k \text{ retrieved domains}\}|} \tag{4.1}$$

$$\text{recall@}k = \frac{|\{k \text{ retrieved domains}\} \cap \{\text{relevant domains}\}|}{|\{\text{relevant domains}\}|} \tag{4.2}$$

In the above equations, the `k retrieved domains` are the domains with the most similar embeddings. The `relevant domains` is the number of domains that share a Symantec label with the query domain. Also, both metrics depend on the cut-off rank; i.e., the number of items that are retrieved by the query. In experiment A, we retrieve 10 items (i.e., $k = 10$) for each query. Precision aims to measure how relevant the retrieved documents are. Recall, instead, measures how many of the relevant documents are retrieved by the query. Note

[3] https://sitereview.bluecoat.com/#/category-test

that the latter is almost always low given the relatively low cut-off rank. Given a Symantec category containing 1000 domains, and a query domain with only that Symantec label, the recall (with $k = 10$) can at most be $10/1000 = 0.01$ even when all retrieved documents were relevant.

In order to get a final score for a set of embeddings, the mean value of all query domains is calculated for both metrics.

4.2.2 Results

Before evaluation of the results, a brief discussion must be dedicated to using the Symantec labels as ground truth. The first consideration is that the labelling is made with a different goal in mind: the activity through a connection might deviate from the categorization of the labels. The second consideration is that finding similar domains is a different activity than making hard categorizations. For example, classifying `medium.com` as website for news articles seems as a valid choice. However, as the website is frequently used in the developer community, it might be more similar to developer domains than to online news platforms. The goal is therefore not to optimize the scoring on Symantec labels. However, the labels do provide some external indication of the quality of the found embeddings for the given domains.

To make sure that the results are measured under equal conditions, the set of testing domains is similar for all entries. Especially recall is heavily affected by the size of the set of domains. Given that $|\{\text{relevant domains}\}| >> k$, a reduction of $|\{\text{relevant domains}\}|$ has an increasing effect on the recall that is nearly linear with reduction of $|\{\text{relevant domains}\}|$. As models with higher minimal occurrence constraints have a smaller number of domains, their score would unfairly be higher if not corrected for the effect. The set of domains that is used for the all experiments is the subset of domains for which an embedding exist for in all models. That is, the domains that occur at least 50 times.

		Precision@10						Recall@10 (e−3)					
	Embedding	5	10	25	50	75	100	5	10	25	50	75	100
window	Skip gram												
1	0	0.165	0.200	0.219	0.221	0.219	0.218	0.559	0.803	0.979	1.000	0.995	0.993
	1	0.170	0.208	0.223	0.222	0.220	0.218	0.595	0.828	0.997	1.008	1.011	1.011
2	0	0.166	0.202	0.222	0.223	0.223	0.222	0.542	0.818	0.982	1.001	1.008	1.009
	1	0.171	0.208	0.225	0.225	0.223	0.221	0.594	0.840	0.999	1.024	1.021	1.016
5	0	0.164	0.203	0.225	0.224	0.224	0.223	0.509	0.811	0.980	1.002	1.003	1.005
	1	0.171	0.210	0.227	0.227	0.225	0.224	0.598	0.840	0.998	1.023	1.025	1.026
10	0	0.160	0.201	0.221	0.221	0.221	0.220	0.480	0.793	0.944	0.975	0.980	0.974
	1	0.172	0.208	0.227	0.227	0.226	0.223	0.601	0.839	0.998	1.019	1.025	1.014
25	0	0.151	0.193	0.213	0.214	0.213	0.212	0.423	0.746	0.899	0.925	0.926	0.920
	1	0.170	0.204	0.222	0.223	0.224	0.223	0.582	0.810	0.966	0.993	1.002	1.008
50	0	0.141	0.182	0.205	0.207	0.207	0.205	0.376	0.677	0.855	0.874	0.883	0.876
	1	0.163	0.199	0.218	0.220	0.221	0.221	0.523	0.778	0.928	0.973	0.985	0.991

Table 4.2: The effect of the embedding size, Architecture and window on the performance of the embeddings.

In Table 4.2 and Table 4.3 the results of the Word2vec embeddings generated with varying parameters are shown. In Table 4.3 it can clearly be seen that given the equal testing sets, no substantial difference can be observed from the minimal occurrence. Even the most infrequent domains (occurrence 1) have no negative effect on the embedding quality. This shows that the Word2vec embeddings are robust against the noise that is brought by these domains. On the other hand, there is also no substantial benefits from the domains that could reveal the more subtle differences among use of internet domains.

Minimal occurrence	Precision@10	Recall@10 (e−3)
1	0.208	0.872
2	0.208	0.876
5	0.208	0.879
10	0.208	0.877
25	0.208	0.874
50	0.207	0.873

Table 4.3: The effect of the minimal occurence on the performance of the w2v embeddings.

In Table 4.2 several interesting patterns can be observed. Firstly, skip gram outperforms the Continuous Bag of Words implementation in almost each direct comparison. Given that the former is usually better in environments where words are more semantically related to each other while the latter performs better for grammatically dependent words, the results show that the most important gain is actually learned from the domains' meaning. Another interesting observation is that the embedding space and window size are highly correlated. A larger window requires more embedding space to account for the increased complexity. So when large window sizes are used, larger embedding spaces always have a positive effect. While the complexity seems to provide a benefit for the recall (i.e., the best scores are found with the highest embedding spaces), the precision does not benefit from it. Given that precision is the most important metric - i.e., the application is directly impacted by the domains that are proposed - an embedding space of 25 is considered sufficient; though 50 is better. Furthermore, a window of 5 and the skip gram architecture is chosen for next experiment.

In Table 4.4 the Word2vec embeddings are tested against baseline embeddings. It can be seen that the baseline is significantly outperformed. Specifically when the relative differences compared to the random model are taken into account, the Word2vec embeddings are considerably better.

Embedding Type	Precision	Recall@10 (e−3)
random	0.089	0.148
Conventional	0.094	0.157
Word2vec	0.227	0.998

Table 4.4: Connection embeddings compared to conventional and random method.

Chapter 5

Evaluation of Embedded Clustering by Intent

This chapter discusses the evaluation of ECBI. The evaluation has been done with the same network traffic data from the Dutch police as used in chapter 4. First, a high-level protocol is set up that is suitable for this thesis project.

5.1 Evaluation protocol

Since network traffic is large in size and open source datasets do not exist due to privacy concerns, there is no labeled data. For this reason, the results of the proposed method cannot be evaluated using traditional machine learning metrics. For that reason, a custom evaluation setup has been set up including expert interviews and technical experiments.

Expert interviews. Since the proposed method is supposed to find clusters given the feedback provided by the analysts, it's ability to do so can only be evaluated by the analysts themselves. If the method is able to provide clusters that are expected according to their expertise, the 'goal' has been achieved. Conversely, if the analysts do not believe that the clusters produced meet their expectations, the "goal" has not been achieved.

During the interviews, analysts are asked to use the method and to compare its use with other methods. By providing comparative methods, the results can be evaluated. Since there were no methods that were tailored to this specific application, two methods that were considered during the course of the thesis project are chosen. While a variety of methods could have been chosen for comparison, it would not have been feasible to include every method. The methods included are *Embedded Topic Modeling* [20] and CBI on which this project is based. The latter could easily be implemented in the tool developed for the thesis project; the only real difference is how the features (and thus the clusters created based on them) are selected. Therefore, all the visualization details and the general workflow are exactly the same. A direct comparison is therefore quite easy. Embedded Topic Modeling, on the other hand, works quite differently. The main reason for this is that there is no real interaction with the analyst on which the model can improve itself. In order to provide a user interface for the analyst to analyze the results of Topic Modeling, *LDAvis* [59] has been implemented.

Before the analysts use the tools, a brief explanation of the methods is provided so that they have enough information to understand how the tool works and what the effect of the provided interaction is. The central question during the interviews is whether the clusters (or topics) found by each method group sessions around features that actually reveal something about the activity in which the entity was engaged. This means that the clusters should not form clusters around, for example, the operating systems used on the entities' machines. With this question in mind, the analysts uses the application. After using the application, questions are asked about their general impression of the application.

Technical Experiments. In addition to the user study, some technical experiments are performed. These experiments test the application in different settings with different parameters. Since there are no labels for the data, this analysis is not able to determine how meaningful the clusters created are. However, the experiments are able to provide insight into how the created clusters differ on quantitatively measurable metrics. *Embedded Topic Modeling* [20] was included in the user study to compare the ability to find insights with the proposed method. However, it is not included in the technical experiments because the two methods are not quantitatively comparable. CBI is compared.

5.2 Expert interviews for the final application

This section is about the interviews with the experts on how well the method proposed in this thesis document performs compared to Topic Modelling and Clustering by Intent.

5.2.1 Setup expert interviews

Interviews are conducted with four experts, all working in different teams that deal with network traffic data. Two of the experts interviewed are mainly involved with the data in a technical way. They, hereafter referred to as technical experts, are responsible for activities such as intercepting the network traffic or transforming the traffic into a structured format. These experts have a deep understanding of the technical aspects of the data, but have limited information about how certain websites are being used for criminal purposes. The other two experts work with the data on a more operational level to investigate criminal behavior. These experts, hereafter referred to as domain experts, are very knowledgeable about specific entities or modus operandi in the dataset, but have less technical understanding.

Each expert is asked to participate in an interview session using the following format:

1. Introduction to the interview and signing informed consent form.
2. Evaluation Embedded Topic Modelling

   (a) Explanation about the method
   (b) Using the application
   (c) Structured questions

3. Evaluation Embedded Clustering by Intent
   (a) Explanation about the method
   (b) Using the application
   (c) Structured questions
4. Evaluation Clustering by Intent
   (a) Explanation about the method
   (b) Using the application
   (c) Structured questions

For each of the applications, about 30 minutes were used to capture the overall impression of each expert. 18 minutes of this time (timed) was used to let the experts experiment with the application. In order to guide the experts and better compare their answers, a specific analysis goal was placed in the center: to find clusters around features that reveal something about the activity of the entity during the session. The rest of the time was used to introduce them to the application and then to ask a series of questions. The following questions were asked:

1. To what extend is the technique able to create clusters of sessions around meaningful attributes that determine what "goal" the entity had in mind?
2. To what extent could the technique support the analysis by providing insights that can be used for further investigation?
3. To what extent are you enabled by the application to determine whether the proposed clusters are meaningful?
4. What are negative aspects of the application?
5. What are positive aspects of the application?
6. How do you compare the clusters created by ECBI compared to the clusters created by CBI?[1]
7. Which of the 3 applications provides the most value for network traffic analysis?[2]

Questions 1-3 and 6-7 were asked with the specific goal of getting the expert to think about things relevant to this thesis project. Questions 4 and 5 were added to ensure that experts are able to provide feedback that may not fit into the first three questions.

[1] This question was asked once after both CBI and ECBI have been evaluated.

[2] This question was asked once after all applications have been evaluated.

Similar pre-processed datasets were used to best compare the tools. CDN domains have been removed[3]from the data and sessions were created with a `session gap time` of 60 minutes. Furthermore, to increase the chances that experts look at domains that are familiar to them, domains were only included if at least 10 entities were connected to them.The Word2vec model for the ECBI method was trained according to the best parameters adopted from the experiment in section 4.2: a skip-gram architecture, an embedding space of 50, and a window of 5. In order to reduce the number of options for the expert, the query parameters were not configurable and instead fixed to $n = 3$, $m = 1$ and $y = 0.1$. The latter means that a connection in a session is only considered if its value (the min-max scaled number of occurrences) is at least 0.1. In other words, a session should contain at least 10% of the occurrences that are in the session with the highest number of occurrences. Both CBI and ECBI use the same set of preset clusters to make the comparison between them as equal as possible. An important configuration for the ETM method was to include only domains that were present in a maximum of 50% of the sessions.

5.2.2 Results

In order to structure the feedback provided by the experts, a categorization was made as follows. First, feedback related to the topic of modeling application is discussed. Then, all feedback related to both ECBI and CBI is discussed together first, as most of the feedback given by the experts was focused on aspects present in both. After that, the differences between the two methods are discussed in a separate section. Finally, the differences between ECBI/CBI and Topic Modeling are discussed before some general feedback on the experimental setup is discussed.

The experts mentioned that they found the user interface (i.e., *LDAvis*) of the ETM application very intuitive. This enabled them to spend the full amount of time to look at actual clusters without the need to discover the tool first. They also liked the fact that they could hover over different clusters. Feedback on the quality of the clusters varied greatly from cluster to cluster. One domain expert mentioned that some clusters had a clear orientation, while other clusters showed a lot of different domains with no direct connection. One of the clusters with a clear orientation was a cluster centered around a specific country - i.e., the domains shared the same top-level domain - which might be interesting, but was not part of the goal of the exercise. Two of the experts mentioned that a good application of the ETM tool would be to find domains when some of the domains within a cluster are already known. Also, one domain expert mentioned a benefit of this application where searches can be initiated by examining related traffic rather than just targeting specific entities or traffic. The tool also facilitates the discovery of an overview of the existing topics within the data. Both technical experts mentioned that it would be good to add functionality that allows a closer look at specific sessions. One of the domain experts wanted to add more information to the interface to help make more informed decisions. The expert mentioned Symantec labels for each domain and entity information for each

[3] Due to a dependency on the Symantec labels, not all CDN domains could be removed. If a domain was not on the Symantec list held by the Dutch police at the time of the experiment, it could not be detected as such.

cluster. The latter was also mentioned by one of the technical experts.

Both ECBI and CBI were difficult for the experts to grasp immediately. All experts mentioned that the methods had a steep learning curve. They struggled to understand how exactly their feedback was being reflected in the machine learning model. For example, one of the technical experts questioned whether a cluster of irrelevant sessions could be added to ensure that those items would not be returned. In addition, finding a good strategy for evaluating a proposed cluster (i.e., what information screens to look at) was something that was not immediately clear. What the experts liked about the (E)CBI approaches is that you are forced to look at a particular cluster. This means that time is spent looking at a pattern that might not be looked at if not necessary. One of the technical experts noted that even if one were to become familiar with the method, these methods would take a considerable amount of time to get to very precise clusters. The expert questioned how the tool should be used; as a tool for an initial search (and thus spend little time on feedback) or to provide more extensive feedback, which also requires more time. One domain expert mentioned that instead of positive filtering (i.e., filtering the domains of interest), these methods require negative filtering (i.e., removing the domains of no interest), which can sometimes be annoying but helps to make the data search more data-driven. According to the expert, this aspect of the model also makes it more difficult to infer familiar clusters. One of the technical experts mentioned that while the visualizations provided reasonable insights into the content of the clusters, the expert would prefer to see a more detailed visualization of the individual sessions.

All experts experienced a steep learning curve when using (E)CBI methods. They could not definitively determine their preference between ECBI and CBI because they felt more familiar with the general concept when using CBI, which contributed to a more positive experience. One of the technical experts liked CBI better because it would result in more specific clusters. These clusters would be more distinctive for a particular activity of the entities. One of the domain experts had the same experience but did not see it as a positive aspect. The expert recognized a set of domains; not as a cluster describing a particular activity, but rather as domains that were very specific to a single entity. The expert mentioned that this would not be the preferred behavior. If a single entity needed to be distinguished, a more thorough analysis using a database lookup on that particular entity would be more efficient.

While the differences between ETM and (E)CBI were easier to specify, the experts did not necessarily have a favorite tool. They mainly argued that the tools have different use cases. ETM would be a good tool if some information about a certain modus operandi (MO) was already known. In that case, ETM could be used to find other domains that might also be related to that MO. On the other hand, (E)CBI would be an appropriate tool to find a new MO for which there is no knowledge yet. Another advantage of ETM compared to (E)CBI is that it takes less time: insights are presented immediately without the need to interact with the underlying machine learning model. With (E)CBI, the likelihood of dropout increases due to the additional time required. ETM was also considered more intuitive by the experts. However, it was also mentioned that ETM favors information bias, as the large number of domains shown leads the expert to domains that are already familiar. By looking at a single cluster at a time, as is the case with (E)CBI, the information bias could be avoided. While some of the experts mentioned that it can be somewhat frustrating to look at clusters that do not directly correspond to insights gained in previous investigations, it provides the opportunity to find new MOs. There were two experts who expressed an explicit preference. One domain expert found ETM to be the preferred tool because it is more intuitive and fits better with their current way of working. The expert noted that this preference might be influenced by the challenging learning process of (E)CBI, which might change over time. One technical expert preferred the (E)CBI methods because they examined each cluster individually, making it more likely to find interesting patterns that had not been seen before.

While the experts were able to experiment with the tools to get an idea, they expressed some concerns about the experimental setup. Particularly for the technical experts, but also for the domain experts, it was often difficult to specify whether a cluster made sense because they had limited information about the domains. To a lesser extent, this could have been addressed by increasing the time allotted for each method. However, building knowledge about how Web sites are used by entities in the data may require an investigation that spans longer periods of time. For example, a cluster that one expert rejected as regular traffic was accepted by another expert based on knowledge gained during a specific investigation.

5.3 Technical experiments for the final application

In this section, technical experiments are performed to measure properties of created clusters. The results of these experiments help to clarify observations made by experts in Section 5.2 with quantitative insights. This section exists 3 experiments where each focusses on a different aspect. The first experiment focusses on the effect of chosen parameters on the creation of the clusters. This experiment has been set up to evaluate the activity dimension. The other two experiments are used to assess the evolution dimension. One of them is about the effect of the number of sessions included in the dataset. The other assesses the impact of session size on the clustering outcomes.

5.3.1 Setup experiment

All experiments evaluate the clusters over a span of 40 active clustering cycles. A user interaction strategy has been fixated on an alternated pattern of rejecting and adding a cluster. So in each even numbered cycle, a cluster is added. For each added cluster, 4 metrics are used to reveal some characteristics:

- **# Entities.** The number of entities reveals something about the generality of a cluster. If a cluster consists of sessions of many different entities, a more general pattern is observed. If instead, in the opposite scenario, the cluster is only about a single entity, the pattern could be specific to the use of that entity.

- **# Features.** While the number of features is defined a priori as a query parameter for the ECBI models, it is dynamic in the CBI model. In case of CBI, a high number of features could be an indication of a more specific pattern while a low (or single) feature is not. For ECBI, this number is determined through the selected parameter.
- **# Sessions.** The number of sessions determines how big the cluster is. A bigger cluster could indicate that the pattern is more common while a small cluster could be an indication of a pattern that only occurred rarely.
- **Precision.** The precision is the score that determines how distinct the cluster is compared to the clusters that were added before. A precision higher than 0.5 indicates that the pattern is more frequent in the newly found cluster while a pattern lower than 0.5 indicates that the pattern is not indicative for the new cluster as it is more frequent in the training data.

In the first experiment the same dataset is used as the dataset used in Section 5.2: CDN domains are removed, domains must be connected to by at least 10 entities and the `session gap time` was set to 60 minutes. The details of this resulting base dataset can be seen in Table 5.1. Furthermore, for each model, the same preset clusters were used as in the interview sessions with the experts. The following combinations of ECBI models were used: 1. $\{n = 3, m = 1\}$, 2. $\{n = 3, m = 2\}$ and 3. $\{n = 5, m = 2\}$. The ECBI models are compared against the conventional CBI method. Each ECBI configuration is tested with $y = 0.1$ as well as with $y = 0.01$.

	Counts			Avg size
Dataset	Connections	Sessions	Entities	Session
Base dataset	16,605,995	64,002	741	259.5
Modification exp. 2a	2,731,124	10,000	629	273.1
Modification exp. 2b	7,363,062	30,000	698	245.4
Modification exp. 3a	16,605,995	249,034	741	66.7
Modification exp. 3b	16,605,995	101,922	741	162.9

Table 5.1: Descriptive statistics about the datasets used in the experiments.

The second experiment, two modifications of the base dataset have been made to simulate two situations where a lower number of sessions is available. The modified datasets include $10,000$ and $30,000$ randomly selected sessions respectively. As there is a smaller focus on the parameters, models are only run with $y = 0.01$ to reduce the number of total models. Details about these two modified datasets can be found as 2a and 2b in Table 5.1.

In the third experiment, two datasets have been created by lowering the `session gap time` to 10 and 30 minutes respectively. Modifications with regard to session size are created in this way because it yields a natural reduction of the average session size without making arbitrary splits. Statistics about these datasets can be found in Table 5.1 as 3a and 3b.

5.3.2 Results

In the first experiment, of which a summary[4]of the results can be seen in Table 5.2, the effect of ECBI with various configurations is compared with CBI. While all models were able to find a cluster in each cycle, only CBI was able to find queries without overlap with clusters that were added before. The most important reason for CBI to find clusters with better precision is that it is allowed to create more specific queries as session require to have all features. An additional effect of these more specific queries is that the clusters are smaller - i.e., on average ±27 while ECBI creates clusters of ±74-141 sessions - which means that less sessions are added to the training data. As there is some variation in each session, additional sessions in the training data imply that the probability of sampling a new query with presence in the training data increases. This also favors the precision of CBI compared to ECBI. Another reason for CBI to have clusters with a higher precision is that ECBI requires clusters to have at least 50 sessions to ensure generality of a observed pattern. CBI, on the other hand is implemented with a maximal cluster size to find a query as specific as possible. CBI's query generation favors the obtained precision compared to ECBI because the query generation is solely focussed on the precision metric. It adds independent features selected on how they distinguish the cluster from the training data. ECBI, instead, constructs the query based on an external mechanism (i.e., the connection embeddings) first and measures the precision only after.

		# Clusters (out of 20)			Average cluster statistics			
Model	y	Total	Full precision	Single entity	Precision	Sessions	Entities	Features
CBI		20	20	8	1.00	27.65	3.60	2.30
ECBI $\{n = 3, m = 1\}$	0.01	20	4	0	0.79	141.85	18.85	3.00
	0.1	20	7	0	0.84	100.55	16.10	3.00
ECBI $\{n = 3, m = 2\}$	0.01	20	9	1	0.91	91.00	9.00	3.00
	0.1	20	7	1	0.68	70.15	13.35	3.00
ECBI $\{n = 5, m = 2\}$	0.01	20	7	1	0.85	114.05	10.15	5.00
	0.1	20	8	1	0.74	74.25	11.80	5.00

Table 5.2: Summary of experiment 1

Because of CBI's specific queries, it generates clusters that are not only smaller, but also contain fewer unique entities. In particular, there are a large number of clusters that have only a single entity, which could indicate that the cluster is more likely to be a combination of domains specific to that entity rather than a more general activity. It can also be seen that ECBI models with a more rigid configuration have a lower number of entities. However, the clusters are very unlikely to have a very low value there. This can be seen from the fact that there is at most one cluster with a single entity.

The direct effect of increasing the y parameter is that a cluster requires a higher presence of a feature to be selected. For this reason, there are fewer sessions that adhere to a query even though the features remain the same. This effect is also reflected in the average number of sessions, which is lower in each ECBI configuration. Since the above effect of the y parameter works for both the training and the residual set, the probability of selecting some sessions in the training data

[4] The full results of all three experiments can be found in Appendix A.

also increases. For this reason, in some cases, relaxing the query (i.e., reducing y) has a negative effect on precision. In other cases, however, relaxing the query may include sessions in the residual set that would not otherwise have met the query criteria. In the case of the rather flexible $\{n = 3, m = 1\}$ configuration, increasing y has a positive effect on precision, while the opposite is true for the other two somewhat stricter configurations. This could be due to the parabolic nature of query strictness. If a query is very accepting, it would be hard to find a unique cluster because it would likely be present in the training data. However, if the query is very specific, it would be hard to find clusters that have the minimum number of sessions. The right balance between them would give the highest precision. This explains the increase in precision for the $\{n = 3, m = 1\}$ configuration, while a decrease is observed for the other two.

		# Clusters (out of 20)			Average cluster statistics			
	Model	Total	Full precision	Single entity	Precision	Sessions	Entities	Features
df 2a	CBI	20	20	1	1.00	18.45	7.30	1.05
	ECBI $\{n = 3, m = 1\}$	15	1	0	0.48	76.00	34.40	3.00
	ECBI $\{n = 3, m = 2\}$	3	0	0	0.39	81.00	41.33	3.00
	ECBI $\{n = 5, m = 2\}$	5	0	0	0.39	81.20	37.40	5.00
df 2b	CBI	20	20	2	1.00	29.45	7.80	1.30
	ECBI $\{n = 3, m = 1\}$	20	4	0	0.70	75.50	18.85	3.00
	ECBI $\{n = 3, m = 2\}$	14	3	0	0.52	68.29	19.36	3.00
	ECBI $\{n = 5, m = 2\}$	15	4	0	0.54	72.27	20.93	5.00

Table 5.3: Summary of experiment 2. All ECBI models are configured with $y = 0.01$

The second experiment, shown in Table 5.3, examines the effect of reducing the amount of data. Especially for the data set where only 10000 sessions are selected (i.e. df 2a), but also for the other data sets, the performance of ECBI is not good. In 2a, at most 1 cluster could be found with full precision. In the case of the $\{n = 3, m = 2\}$ configuration, no more than a total of 3 clusters could be found with an average precision of only 0.39. The overall poor performance can be partly explained by the fact that the minimum session threshold of 50 is too restrictive for the overall data size. Since there is 2-6 times less data than in the baseline dataset, it is much harder to find clusters with more than 50 sessions. This is illustrated by the fact that the average cluster in the baseline experiment contained ±70-140 sessions. Another interesting observation is that due to the smaller data size, CBI could often find a cluster with only a single feature. This led to a significant increase in the average number of entities included in the clusters.

In the third experiment, the models are executed on data with smaller sessions. A summary of the results is shown in Table 5.4. An important side effect of reducing the average number of connections per session is that the number of sessions increases accordingly. This has implications for both CBI and ECBI. The former requires relatively more features to get below the threshold, while the latter can reach the minimum number of sessions more easily. This effect can be seen in the results. The ECBI methods perform slightly better in terms of precision, and the feature statistic has increased for CBI. The increase in features also further reduced the number of entities per cluster, making it even more specific than in the baseline experiment.

		# Clusters (out of 20)			Average cluster statistics			
	Model	Total	Full precision	Single entity	Precision	Sessions	Entities	Features
df 3a	CBI	20	20	11	1.00	30.60	3.15	2.35
	ECBI $\{n = 3, m = 1\}$	20	10	1	0.89	144.40	15.25	3.00
	ECBI $\{n = 3, m = 2\}$	20	13	4	0.87	87.00	8.00	3.00
	ECBI $\{n = 5, m = 2\}$	20	15	2	0.92	91.90	10.00	5.00
df 3b	CBI	20	20	15	1.00	22.15	1.60	2.45
	ECBI $\{n = 3, m = 1\}$	20	10	0	0.87	109.60	10.10	3.00
	ECBI $\{n = 3, m = 2\}$	20	9	3	0.78	70.00	9.40	3.00
	ECBI $\{n = 5, m = 2\}$	20	10	2	0.80	82.45	9.65	5.00

Table 5.4: Summary of experiment 3. All ECBI models are configured with $y = 0.01$

Dataset	# unique domains	Session size
Session `SG=60` (base)	27.69	259.5
Session `SG=10` (df 3a)	10.40	66.7
Session `SG=30` (df 3b)	19.38	162.9

Table 5.5: Variety for sessions with different values of `session gap time` (`SG`)

More interesting is the difference in performance between different configurations of ECBI compared to the baseline experiment. Looking at the baseline experiment set up with $y = 0.01$, the $\{n = 3, m = 2\}$ configuration outperformed the other two configurations with a delta of more than 0.05 in precision. In this experiment it can be seen that the $\{n = 5, m = 2\}$ configuration is as good in 3b and even better in 3a. This can be explained by the fact that the number of unique features in each session has decreased as a result of the reduction in total connections. This reduction, quantified in Table 5.5, has two effects. (1) The reduced variety in sessions makes it less likely that there are irrelevant features - to the main features on which it was originally clustered - that cause overlap with the newly proposed cluster. (2) On the other hand, features that are related and that appeared in the session before are now also likely to appear in separate sessions. The first reason is beneficial because compared to the baseline experiment, $\{n = 5, m = 2\}$ had a higher change to affect more sessions because it has a higher value of n. The second reason benefits the configuration because it has a total of 5 features that it can use to connect sessions that a different configuration with a smaller n cannot. This means that smaller session sizes allow the ECBI models to make more use of the connection embeddings to find synonym features.

Chapter 6

Discussion

This section has three main objectives. First, it synthesizes the different evaluation methods to provide a comprehensive review of the method from different perspectives. Based on this review, observed peculiarities are discussed and directions for future work are presented.

The experts did not give a clear judgment in the direction of one of the clustering methods. Rather, they indicated a specific purpose for (E)CBI on the one hand and ETM on the other hand. ETM, which was perceived as more intuitive by the experts, provides an overview of different topics without the need for feedback from the expert. However, some experts noted that the topics are not necessarily focused on activities; a topic with domains from a specific national top-level domain was observed. Given that ETM does not allow for expert feedback and that network traffic can be clustered in many different ways, this is a problem that is difficult to avoid. ETM presents a more or less static ordering of data that may or may not match the intent of the expert, but cannot use external knowledge. (E)CBI, on the other hand, is able to learn from feedback, allowing the expert to influence how clusters are generated.

Another major difference between the models is that ETM presents all topics in a single window, while (E)CBI requires the analyst to iteratively go through the data and only then have a summary of the results. The experts mentioned that ETM allowed them to look at phenomena they were already familiar with, which led to more recognition, i.e. less attention was paid to unknown areas. (E)CBI, on the other hand, forced the experts to look at the clusters one at a time. While this takes more time, the experts mentioned that they look at clusters that they might not have looked at if they could move immediately to the next cluster. This could reduce human bias and lead to new insights.

For the two reasons mentioned above, ETM would probably be a better tool for an ongoing investigation with a pre-determined topic. In this scenario, the ability of ETM to zoom in on sessions around predetermined aspects of the data could be very convenient. However, in a more exploratory phase, (E)CBI can add a lot of value by focusing on one cluster at a time. This obligation to provide feedback in the (E)CBI methods could be an advantage, i.e. the experts need to look more closely at some of the clusters that would not have been checked otherwise.

Though, one of the experts mentioned that while some of the domains did not form a cluster together, the individual domains could indicate interesting behavior. Within the current set of feedback options, such feedback cannot be given. Therefore, future work could explore additional or different feedback options for the experts that better fit the complex analysis of the data.

Because internet sessions might contain different kinds of traffic, finding clusters around a particular topic is very difficult. Before ECBI was implemented at the session level, it was developed to cluster entities directly. By selecting two initial clusters, the models (either CBI or ECBI) were already disabled to find a query without overlapping with existing clusters. While sessions have far fewer unique domains, the lack of consistency remains a challenge. Reducing the session size, and thus reducing the diversity, further isolates the entity's primary activity.

Compared to ECBI, CBI creates more specific clusters. This is due to fact that sessions are required to contain all the features. One of the experts preferred CBI over ECBI because of this. Though, a domain expert mentioned that this more specific behavior allowed to cluster around entity specific features rather than features describing a particular activity. The latter was also confirmed by the technical experiments. CBI has a tendency to generate clusters around one (or very few) entities. This would be hard to prevent, since CBI only optimizes for good precision scores and selects features based on this criterion only. However, for ECBI, which generates queries independent of the precision criterion, it may be possible to find activity-specific clusters when performed with these reduced session sizes.

Adjusting the query parameters for ECBI every cycle is something that could not be evaluated. For the expert interviews, this would introduce an additional level of freedom that would make it difficult to compare individual interviews. The same is true for the technical experiments. There would be many more additional strategies, making it difficult to reason about the configurations themselves. However, additional flexibility in the configuring the query parameters would allow for finding specific clusters at the beginning, when there is a lot of data available, while being more flexible later, when the search space becomes more restricted. While this could theoretically benefit the results, making the user responsible for going through a large number of different configurations in each cycle would increase the workload significantly.

Both the effect of the session size and experimenting with more advanced query strategies would be interesting directions for future work. The effect of reduced session sizes could be studied more closely, either by using smaller values of `session gap time` or by implementing more sophisticated separation techniques that better extract the activity. Automating the process of selecting a query that allows multiple query parameters would also be interesting to explore. However, this would require a better optimization metric, not just precision and cluster size.

The visualization of the data was considered an important part of this thesis project. For this reason, the information screens needed to make informed decisions were discussed with relevant people. The effect of these efforts is also reflected in the expert interviews. Information that was missing in *LDAvis*, a tool not intended for network traffic analysis, was included in the tool for (E)CBI. However, one expert mentioned that more information, the Symantec label of a domain, would speed up and improve decision making. The main focus of this paper was to find visualizations that display the data itself in a meaningful way. This could be sufficient for textual documents, where experts that speak the language have sufficient information by reading the textual documents themselves. Yet, given the complex landscape where knowledge is fragmented, it may be beneficial for future studies to look at enriching network traffic with external datasets.

Evaluating the clusters of network traffic is very difficult. The in this thesis project allotted time of 1.5 hours, was a limiting factor. Especially for the steep learning curve of (E)CBI methods. The experts mentioned that more time would have helped them to understand the tools better and to spend more time consulting external resources. This ultimately impacted the judgement of the tools given that CBI was assessed before CBI in each interview session. Given the few interviewees, this factor could not have been avoided.

However, even if more time was alloted, it might not have been enough. Experts only have knowledge in a limited number of areas due to the need for specialization given the large domain. For this reason, a cluster found interesting by one expert was rejected by another. Unlike a language, domain expertise about the full network traffic landscape is not something which can reasonably be expected from domain experts to possess. For that reason, it would not have been possible to allocate a reasonable amount of time to the interviews. In order to obtain more reliable results, a solution would be to let domain experts use the tool over a longer period of time as a part of their ongoing activities. However, this raises another problem. In order for experts to use the tool for ongoing work and dive deeper into proposed clusters, a certain level of trust in the system is required. This trust is needed to see clusters as potential new insights rather than irrelevant patterns that do not fit their experience.

The technical experiments could partly solve this problem as they are not subjective to human limitations. They can provide guidance in quantifying certain characteristics of the data. However, they cannot determine the quality of the clustering results; a larger or more specific cluster does not define its quality. While it is tempting to qualify results with a higher precision score as better, it cannot be used as such. If queries were created specifically for a single session, they would always find a full-precision ‘cluster’, but it would have no value at all.

The challenge regarding the evaluation of the tool offers several opportunities for future work. First, engaging experts over a longer period of time, so that the clustering results can be followed up with in-depth investigations, could provide a more reliable evaluation. However, it might be even more valuable to explore ways to solve problems underlying the evaluation problem. For example, given that the scattered knowledge is inherent to the network traffic landscape, exploring methods to combine feedback from several experts is worth looking into.

As no work had been found on clustering network traffic, this thesis project adapted text mining methods. This choice had been made because similarities were observed in the characteristics of the data. The basic element of both data types - i.e., words and connections - are both sequential and discrete. However, an important difference does also exist. Network traffic, compared to textual data, seem to allow more options to express oneself. This means that individuals have more opportunities to perform actions on the internet than may exist in language. An activity could be a composition of multiple individual connections and for all of these individual connections multiple options exist.

An illustration of such composition is where an entity books a vacation by first going to the website of a travel agency followed by a redirect to their payment method's website to settle the transaction. While all combinations of travel agencies and payment methods would probably indicate the same activity, a single combination is mostly related to the preference (i.e., the setup) of a single entity. The implications of this phenomenon can be observed in the data through the CBI clusters. Most clusters created by CBI that are based on more than a single feature contain very few or even a single entity only. For ECBI, that extends the reach of a query to several synonym connections, chances are reduced that such specific setup is selected. Using the parameters m and n the right balance can be obtained to allow synonyms while making sure that the query remains specific as well.

Chapter 7

Conclusion

In this thesis project, we set forward the research question *How to actively cluster network traffic based on an analyst's intent?*. Since no relevant studies could be found within the domain of network traffic analysis, methods from the text mining domain were considered. To answer the research question, a clustering method inspired by CBI has been developed that allows a clustering approach for internet sessions where external expert knowledge can be used. To develop this method, Embedded Clustering By Intent (ECBI), the research question was divided into three smaller sub-questions to facilitate problem solving.

The first subquestion proposed a method for finding a set of features that can meaningfully distinguish a group of sessions. Due to the substantial presence of synonyms in network traffic data, a feature is often only indicative for an activity in a subset of sessions. In order to make the selection of sessions more robust, a group of synonym features could capture a bigger set of these sessions. In order to find synonym features, connection embeddings were created. The evaluation showed that these embeddings were significantly better at identifying feature similarity than conventional methods.

In order to use the feature sets, answering subquestion 2, an interactive screen to gather feedback from analysts was implemented. However, since the analysis of internet traffic is not obvious, the third question explored ways to visualize the data. This resulted in three information screens: a window with high level information, the Shifterator plot and a window to zoom in on individual sessions.

We evaluated ECBI on specifically sampled intercepted network traffic from the Dutch National Police. The result of all research questions, the network clustering tool ECBI, was evaluated on specifically sampled intercepted network traffic from the Dutch National Police. In this evaluation, ECBI was compared against ETM and CBI using expert interviews and technical experiments. Rather than a single 'best' tool, the expert interviews showed that both ETM and (E)CBI have their own purpose. ETM would be more appropriate when searching for specific information. (E)CBI, on the other hand, could provide insights into criminal activities that have not been found before. However, it was observed that clusters created by ECBI contained clusters of substantial more entities than clusters of CBI. This indicates that instead of finding features specific to an entity, ECBI found a set of features that revealed a more general pattern; a possible activity.

Clustering network traffic is a challenging task due to the complexity of evaluating the data and the incoherent nature of internet sessions. However, we demonstrated that it is possible to create connection embeddings to identify similar features in network traffic. Based on these embeddings, an active method called ECBI was developed to cluster the internet sessions accordingly.

Bibliography

[1] Andres M. Grisales A., Sebastian Robledo, and Martha Zuluaga. Topic modeling: Perspectives from a literature review. *IEEE Access*, 11:4066–4078, 2023.

[2] Mahmoud Abbasi, Amin Shahraki, and Amir Taherkordi. Deep Learning for Network Traffic Monitoring and Analysis (NTMA): A Survey. *Computer Communications*, 170(September 2020):19–41, 2021.

[3] Soliman Abd Elmonsef Sarhan, Hassan A. Youness, and Ayman M. Bahaa-Eldin. A framework for digital forensics of encrypted real-time network traffic, instant messaging, and VoIP application case study. *Ain Shams Engineering Journal*, 14(9):102069, sep 2023.

[4] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. *ACM SIGMOD Record*, 28(2):61–72, June 1999.

[5] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. ACM, June 1998.

[6] Pallavi Asrodia and Hemlata Patel. Analysis of various packet sniffing tools for network monitoring and analysis. *International Journal of Electrical, Electronics and Computer Engineering*, 1(1):55–58, 2012.

[7] Pallavi Asrodia and Hemlata Patel. Network traffic analysis using packet sniffer. *International Journal of Engineering Research and Applications (IJERA)*, 2(3):854–856, 2012.

[8] Karel Bartos, Michal Sofka, and Vojtech Franc. Optimized invariant representation of network traffic for detecting unseen malware variants. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 807–822, Austin, TX, August 2016. USENIX Association.

[9] Jay Beale, Angela Orebaugh, and Gilbert Ramirez. *Introducing Wireshark: Network Protocol Analyzer*, chapter 2, pages 51–98. Syngress, 2007.

[10] Benjamin Bergner and Georg Krempl. Active subtopic detection in multi-topic data. In *CEUR Workshop Proceedings*, volume 1707, pages 35 – 44, 2016.

[11] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is “nearest neighbor” meaningful? In *Lecture Notes in Computer Science*, pages 217–235. Springer Berlin Heidelberg, 1999.

[12] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[13] Robert T. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122, October 1989.

[14] Jianghui Cai, Jing Hao, Haifeng Yang, Xujun Zhao, and Yuqing Yang. A review on semi-supervised clustering. *Information Sciences*, 632:164–200, June 2023.

[15] John Carr. A brief history of child safety online, May 2017.

[16] Brian D. Carrier. Defining digital forensic examination and analysis tool using abstraction layers. *Int. J. Digit. EVid.*, 1, 2003.

[17] Ben Carterette and Ellen M. Voorhees. *Overview of Information Retrieval Evaluation*, page 69–85. Springer Berlin Heidelberg, 2011.

[18] V. Cerf and R. Kahn. A Protocol for Packet Network Intercommunication. *IEEE Transactions on Communications*, 22:637–648, 1974.

[19] Tara Chari and Lior Pachter. The specious art of single-cell genomics. *PLOS Computational Biology*, 19(8):e1011288, August 2023.

[20] Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. Topic modeling in embedding spaces. *CoRR*, abs/1907.04907, 2019.

[21] Edsger W. Dijkstra. The structure of the “the”-multiprogramming system. In *Proceedings of the ACM symposium on Operating System Principles - SOSP '67*. ACM Press, 1967.

[22] Daniel Engel, Lars Hüttenberger, and Bernd Hamann. A Survey of Dimension Reduction Methods for High-dimensional Data Analysis and Visualization. In Christoph Garth, Ariane Middel, and Hans Hagen, editors, *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011*, volume 27 of *OpenAccess Series in Informatics (OASIcs)*, pages 135–149, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[23] Kevin R. Fall and Richard Stevens. *TCP/IP Illustrated*. Addison-Wesley, 2 edition, 2012.

[24] Andrey Ferriyan, Achmad Husni Thamrin, Keiji Takeda, and Jun Murai. Encrypted malicious traffic detection based on word2vec. *Electronics*, 11(5):679, February 2022.

[25] George Forman, Hila Nachlieli, and Renato Keshet. Clustering by intent: A semi-supervised method to discover relevant clusters incrementally. In *Machine Learning and Knowledge Discovery in Databases*, pages 20–36. Springer International Publishing, 2015.

[26] Ryan J. Gallagher, Morgan R. Frank, Lewis Mitchell, Aaron J. Schwartz, Andrew J. Reagan, Christopher M. Danforth, and Peter Sheridan Dodds. Generalized word shift graphs: a method for visualizing and explaining pairwise comparisons between texts. *EPJ Data Science*, 10(1), January 2021.

[27] Gérard Govaert and Mohamed Nadif. *Introduction*, pages 15–52. Wiley, December 2013.

[28] Piyush Goyal and Anurag Goyal. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. In *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*, volume 2018-Janua, pages 77–81. IEEE, sep 2017.

[29] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, August 2015.

[30] Andres M. Grisales, Sebastian Robledo, and Martha Zuluaga. Topic modeling: Perspectives from a literature review. *IEEE Access*, 11(November 2022):4066–4078, 2023.

[31] Aditya Grover and Jure Leskovec. node2vec. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, August 2016.

[32] Li Guo, Haitao Gan, Siyu Xia, Xiaobin Xu, and Tao Zhou. Joint exploring of risky labeled and unlabeled samples for safe semi-supervised clustering. *Expert Systems with Applications*, 176:114796, August 2021.

[33] Neal Jean, Sherrie Wang, Anshul Samar, George Azzari, David Lobell, and Stefano Ermon. Tile2vec: Unsupervised representation learning for spatially distributed data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3967–3974, July 2019.

[34] Zhen Jiang, Yongzhao Zhan, Qirong Mao, and Yang Du. Semi-supervised clustering under a compact-cluster assumption. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2022.

[35] Junghoon Kim, Kaiyu Feng, Gao Cong, Diwen Zhu, Wenyuan Yu, and Chunyan Miao. ABC. *Proceedings of the VLDB Endowment*, 15(10):2134–2147, June 2022.

[36] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data. *ACM Transactions on Knowledge Discovery from Data*, 3(1):1–58, March 2009.

[37] Punit Kumar and Atul Gupta. Active learning query strategies for classification, regression, and clustering: A survey. *Journal of Computer Science and Technology*, 35(4):913–945, July 2020.

[38] Satish Kumar, Sunanda Gupta, and Sakshi Arora. Research trends in network-based intrusion detection systems: A review. *IEEE Access*, 9:157761–157779, 2021.

[39] Satish Kumar, Sunanda Gupta, and Sakshi Arora. Research Trends in Network-Based Intrusion Detection Systems: A Review. *IEEE Access*, 9:157761–157779, 2021.

[40] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

[41] Jieling Li, Hao Zhang, and Zhiqiang Wei. The weighted word2vec paragraph vectors for anomaly detection over HTTP traffic. *IEEE Access*, 8:141787–141798, 2020.

[42] Xueting Liao, Danyang Zheng, and Xiaojun Cao. Coronavirus pandemic analysis through tripartite graph clustering in online social networks. *Big Data Mining and Analytics*, 4(4):242–251, December 2021.

[43] Chin Ying Liew, Jane Labadin, Woon Chee Kok, and Monday Okpoto Eze. A methodology framework for bipartite network modeling. *Applied Network Science*, 8(1), January 2023.

[44] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (PCA). *Computers & Geosciences*, 19(3):303–342, March 1993.

[45] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[46] Ric Messier. *Network Forensics*. John Wiley & Sons, 2017.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[48] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[49] Shane Miller, Kevin Curran, and Tom Lunney. Multilayer Perceptron Neural Network for Detection of Encrypted VPN Network Traffic. In *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, pages 1–8. IEEE, jun 2018.

[50] Philip O'Kane, Sakir Sezer, and Domhnall Carlin. Evolution of ransomware. *IET Networks*, 7(5):321–327, September 2018.

[51] Divya Pandove, Shivan Goel, and Rinkl Rani. Systematic review of clustering high-dimensional and large datasets. *ACM Transactions on Knowledge Discovery from Data*, 12(2):1–68, January 2018.

[52] Dimitrios Papamartzivanos, Félix Gómez Mármol, and Georgios Kambourakis. Dendron: Genetic trees driven rule induction for network intrusion detection systems. *Future Generation Computer Systems*, 79:558–574, 2018.

[53] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, June 2004.

[54] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, December 1999.

[55] Andrés F. Murillo Piedrahita, Vikram Gaur, Jairo Giraldo, Alvaro A. Cardenas, and Sandra Julieta Rueda. Virtual incident response functions in control systems. *Computer Networks*, 135:147–159, apr 2018.

[56] David C. Pyrooz, Scott H. Decker, and Richard K. Moule. Criminal and routine activities in online settings: Gangs, offenders, and the internet. *Justice Quarterly*, 32(3):471–499, March 2013.

[57] Jörg Rahnenführer, Riccardo De Bin, Axel Benner, Federico Ambrogi, Lara Lusa, Anne-Laure Boulesteix, Eugenia Migliavacca, Harald Binder, Stefan Michiels, Willi Sauerbrei, and Lisa McShane. Statistical analysis of high-dimensional biomedical data: a gentle introduction to analytical goals, common approaches and challenges. *BMC Medicine*, 21(1), May 2023.

[58] Christian Rossow, Christian J. Dietrich, Herbert Bos, Lorenzo Cavallaro, Maarten van Steen, Felix C. Freiling, and Norbert Pohlmann. Sandnet. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pages 78–88. ACM, apr 2011.

[59] Carson Sievert and Kenneth Shirley. Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*. Association for Computational Linguistics, 2014.

[60] Leslie F. Sikos. Packet analysis for network forensics: A comprehensive survey. *Forensic Science International: Digital Investigation*, 32:200892, mar 2020.

[61] W. Stallings and R. Van Slyke. *Business Data Communications*. Stallings, William: William Stallings books on computer and data communications technology. Prentice Hall, 2001.

[62] Andrew S. Tanenbaum and David Wetherall. *Computer Networks*. Prentice Hall, 5 edition, 2011.

[63] Alaa Tharwat and Wolfram Schenck. A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820, February 2023.

[64] Princeton University. About wordnet. WordNet. Princeton University, 2010.

[65] Erwin van de Wiel, Mark Scanlon, and Nhien-An Le-Khac. Enabling non-expert analysis OF large volumes OF intercepted network traffic. In *Advances in Digital Forensics XIV*, pages 183–197. Springer International Publishing, 2018.

[66] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[67] Avgoustinos Vouros and Eleni Vasilaki. A semi-supervised sparse k-means algorithm. *Pattern Recognition Letters*, 142:65–71, February 2021.

[68] Meng Wang, Yiqin Lu, and Jiancheng Qin. A dynamic MLP-based DDoS attack detection method using feature selection and feedback. *Computers & Security*, 88:101645, jan 2020.

[69] Juan Xie, Anjun Ma, Anne Fennell, Qin Ma, and Jing Zhao. It is time to apply biclustering: a comprehensive review of biclustering applications in biological and biomedical data. *Briefings in Bioinformatics*, 20(4):1450–1465, February 2018.

[70] Panpan Xu, Nan Cao, Huamin Qu, and John Stasko. Interactive visual co-cluster analysis of bipartite graphs. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, April 2016.

[71] Yan Yang, Wei Tan, Tianrui Li, and Da Ruan. Consensus clustering based on constrained self-organizing map and improved cop-kmeans ensemble in intelligent decision support systems. *Knowledge-Based Systems*, 32:101–115, August 2012.

[72] Juan Zhao, Sachin Shetty, Jan Wei Pan, Charles Kamhoua, and Kevin Kwiat. Transfer learning for detecting unknown network attacks. *EURASIP Journal on Information Security*, 2019(1):1, dec 2019.

[73] Linhong Zhu, Aram Galstyan, James Cheng, and Kristina Lerman. Tripartite graph clustering for dynamic sentiment analysis on social media. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. ACM, June 2014.

[74] H. Zimmermann. Osi reference model - the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.

Appendix A

Full technical experiments results

	Cycle	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
CBI	# Entities	1	1	1	9	1	2	2	6	4	1	7	1	3	1	11	7	2	4	1	7
$y = 0.01$	# Features	4	2	2	2	2	6	2	2	2	2	2	2	2	2	2	3	2	2	2	1
	# Sessions	43	24	17	38	17	42	36	13	45	13	46	29	21	5	24	47	31	4	9	49
	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
ECBI	# Entities	31	4	20	8	17	10	21	20	15	5	34	9	14	13	36	39	14	18	27	22
$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$m = 1$	# Sessions	194	114	90	75	370	65	147	80	61	200	247	144	104	65	451	124	65	58	66	117
$y = 0.01$	Precision	1.00	1.00	1.00	1.00	0.92	0.86	0.83	0.82	0.78	0.78	0.77	0.77	0.72	0.70	0.70	0.66	0.66	0.63	0.62	0.62
ECBI	# Entities	18	18	15	11	20	10	23	20	16	23	7	31	10	5	12	4	12	19	30	18
$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$m = 1$	# Sessions	191	189	131	101	84	63	57	84	71	53	87	169	52	89	82	52	199	88	91	78
$y = 0.1$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.88	0.87	0.85	0.82	0.80	0.77	0.75	0.74	0.72	0.72	0.65	0.62	0.61
ECBI	# Entities	21	5	7	6	13	8	4	2	1	2	3	16	10	4	19	9	16	8	23	3
$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$m = 2$	# Sessions	223	160	138	109	88	78	71	61	57	103	89	64	54	71	68	63	68	83	106	66
$y = 0.01$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.92	0.91	0.89	0.87	0.82	0.82	0.82	0.83	0.86	0.81	0.74
ECBI	# Entities	8	2	43	1	10	2	5	4	15	20	16	12	9	21	20	16	16	9	16	22
$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$m = 2$	# Sessions	116	92	74	68	62	54	51	57	50	68	54	71	67	152	68	50	61	61	55	72
$y = 0.1$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.79	0.78	0.72	0.59	0.55	0.55	0.48	0.43	0.39	0.38	0.32	0.31	0.29
ECBI	# Entities	21	8	18	2	16	7	2	1	15	11	18	6	5	19	15	2	12	5	9	11
$n = 5$	# Features	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
$m = 2$	# Sessions	232	176	132	85	71	65	51	180	79	92	238	97	51	87	64	93	247	70	53	118
$y = 0.01$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94	0.89	0.91	0.85	0.82	0.79	0.72	0.72	0.69	0.66	0.73	0.64	0.63
ECBI	# Entities	6	4	10	4	7	9	8	4	1	12	23	11	10	13	12	15	22	22	5	38
$n = 5$	# Features	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
$m = 2$	# Sessions	115	88	83	87	60	52	57	55	69	50	79	70	71	75	51	70	56	63	59	175
$y = 0.1$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.84	0.80	0.69	0.61	0.63	0.60	0.58	0.46	0.42	0.37	0.36	0.34

Table A.1: Experiment 1 results: effect of parameters on clustering results.

		Cycle	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
Modification exp. 2a	CBI	# Entities	3	15	21	12	10	10	6	10	4	9	6	7	7	5	3	3	7	4	3	1
		# Features	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		# Sessions	11	50	40	31	28	23	18	18	17	15	14	15	13	12	12	11	11	10	10	10
		Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	ECBIN	# Entities	54	18	52	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	$n = 3$	# Features	3	3	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	$m = 2$	# Sessions	96	50	97	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	$y = 0.01$	Precision	0.27	0.78	0.13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	ECBI	# Entities	30	21	14	14	17	48	31	78	34	33	45	21	48	31	51	-	-	-	-	-
	$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	-	-	-	-
	$m = 1$	# Sessions	60	61	55	57	54	95	72	226	58	52	80	56	76	60	78	-	-	-	-	-
	$y = 0.01$	Precision	1	0.77	0.61	0.56	0.61	0.54	0.50	0.49	0.39	0.38	0.35	0.29	0.26	0.24	0.22	-	-	-	-	-
	ECBI	# Entities	21	28	78	24	36	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	$n = 5$	# Features	5	5	5	5	5 0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	$m = 2$	# Sessions	62	54	176	51	63	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	$y = 0.01$	Precision	0.59	0.42	0.46	0.39	0.09	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Modification exp. 2b	CBI	# Entities	10	9	1	7	3	6	7	11	9	2	4	10	10	9	19	11	16	7	1	4
		# Features	2	2	3	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		# Sessions	44	26	9	27	7	50	46	38	38	38	37	33	32	27	26	24	24	22	21	20
		Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	ECBIN	# Entities	31	15	11	5	4	10	34	10	17	24	26	43	19	22	-	-	-	-	-	-
	$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	-	-	-	-	-
	$m = 2$	# Sessions	99	76	63	62	89	50	74	55	63	50	52	83	74	66	-	-	-	-	-	-
	$y = 0.01$	Precision	1.00	1.00	1.00	0.81	0.78	0.61	0.52	0.41	0.27	0.27	0.22	0.17	0.16	0.12	-	-	-	-	-	-
	ECBI	# Entities	41	37	17	15	8	17	15	7	25	17	17	27	16	11	12	16	12	32	13	22
	$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	$m = 1$	# Sessions	135	122	57	84	88	145	53	67	52	93	69	93	52	57	52	52	62	60	67	50
	$y = 0.01$	Precision	1.00	1.00	1.00	0.87	0.81	0.80	1	0.71	0.67	0.63	0.64	0.62	0.58	0.56	0.57	0.56	0.53	0.48	0.44	0.43
	ECBI	# Entities	14	16	8	10	14	17	46	7	19	11	16	61	21	35	19	-	-	-	-	-
	$n = 5$	# Features	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	-	-	-	-	-
	$m = 2$	# Sessions	79	77	55	50	60	51	88	63	56	60	79	134	87	90	55	-	-	-	-	-
	$y = 0.01$	Precision	1.00	1.00	1.00	1.00	0.81	0.81	0.47	0.49	0.35	0.32	0.24	0.23	0.19	0.14	0.12	-	-	-	-	-

Table A.2: Experiment 2 results: impact of sessions count on clustering results.

		Cycle	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
Modification exp. 3a	CBI	# Entities	6	1	2	1	1	1	18	11	4	1	1	1	1	2	2	2	1	5	1	1
		# Features	3	2	2	2	2	2	2	4	3	2	2	2	5	2	2	2	2	2	2	2
		# Sessions	40	45	49	32	21	18	40	38	26	41	21	8	26	43	23	26	13	48	42	12
		Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	ECBI	# Entities	11	1	11	8	5	2	2	5	2	1	14	20	6	18	6	1	29	1	11	6
	$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	$m = 2$	# Sessions	176	130	107	98	89	83	77	72	66	61	52	52	51	102	84	63	157	72	97	51
	$y = 0.01$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.77	0.75	0.70	0.67	0.60	0.45	0.43
	ECBI	# Entities	53	37	18	13	6	3	1	9	7	6	17	20	28	21	10	5	4	21	7	19
	$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	$m = 1$	# Sessions	356	183	164	122	107	85	70	65	59	52	342	142	102	272	259	105	165	80	101	57
	$y = 0.01$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.93	0.86	0.82	0.81	0.78	0.76	0.72	0.72	0.69	0.66
	ECBI	# Entities	9	4	12	1	18	5	6	15	5	17	5	12	10	12	20	2	1	15	3	28
	$n = 5$	# Features	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	$m = 2$	# Sessions	257	128	121	115	102	87	80	97	76	69	66	62	56	52	55	120	77	58	66	94
	$y = 0.01$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.83	0.76	0.72	0.60	0.59
Modification exp. 3b	CBI	# Entities	1	1	1	1	1	4	1	1	2	1	1	2	1	4	1	1	5	1	1	1
		# Features	3	2	3	2	3	4	2	2	2	2	5	2	3	2	2	2	2	2	2	2
		# Sessions	11	18	25	26	46	47	42	11	24	18	24	12	42	21	16	6	37	5	8	4
		Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	ECBI	# Entities	5	5	4	3	8	3	1	3	16	4	1	8	1	15	6	11	51	4	22	17
	$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	$m = 2$	# Sessions	137	112	95	75	72	66	63	57	52	58	54	50	50	53	50	72	73	77	76	58
	$y = 0.01$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.74	0.73	1.00	1.00	0.75	0.76	0.63	0.63	0.56	0.52	0.46	0.41	0.41
	ECBI	# Entities	13	19	23	4	11	2	12	13	13	13	6	2	8	11	5	4	11	7	5	20
	$n = 3$	# Features	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	$m = 1$	# Sessions	257	162	154	101	98	89	82	76	59	57	148	57	51	78	197	103	100	160	68	95
	$y = 0.01$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.85	0.82	0.81	0.77	0.72	0.72	0.68	0.69	0.67	0.63
	ECBI	# Entities	5	1	4	10	11	18	7	10	2	9	1	16	15	13	13	16	5	6	23	8
	$n = 5$	# Features	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	$m = 2$	# Sessions	154	139	127	97	78	70	63	54	60	52	65	52	64	88	76	63	140	54	77	76
	$y = 0.01$	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.80	0.77	0.69	0.68	0.59	0.56	0.51	0.49	0.47	0.45

Table A.3: Experiment 3 results: assessing the impact of session size on clustering outcomes.