Session stitching using sequence fingerprinting for web page visits

Session Stitching Using Sequence Fingerprinting For Web Page Visits

Johannes De Smedt*

Research Centre for Information Systems Engineering, KU Leuven, Leuven, Belgium

Ewelina Lacka

University of Edinburgh Business School, Edinburgh, UK

Spyro Nita

Edinburgh Parallel Computing Centre, Edinburgh, UK

Hans-Helmut Kohls, Ross Paton

QueryClick, Edinburgh, UK

Abstract

The nature of people's web navigation has significantly changed in recent years. The advent of smartphones and other handheld devices has given rise to web users consulting websites with more than one device, or using a shared device. As a result, large volumes of seemingly disjoint data are available, which when analysed together can support decision-making. The task of identifying web sessions by linking such data back to a specific person, however, is hard. The idea of session stitching aims to overcome this by using machine learning inference to identify similar or identical users. Many such efforts use various demographic data or device-based features to train matching algorithms. However, often these variables are not available for every dataset or are recorded differently, making a streamlined setup difficult. Besides, the often result in vast feature spaces which are hard to use for actionable interpretation.

*Corresponding author

Email addresses: johannes.desmedt@kuleuven.be (Johannes De Smedt), ewelina.lacka@ed.ac.uk (Ewelina Lacka), s.nita@epcc.ed.ac.uk (Spyro Nita), {hans;ross}@queryclick.com (Hans-Helmut Kohls, Ross Paton)

In this paper, we present an alternative approach based on the fingerprinting of web pages visited by users in a single session. By learning behavioral patterns from these sequences of page visits, we obtain features that can be used for matching without requiring sensitive user-agent data such as IP, geo location, or device details as is common with other approaches. Using these sequential fingerprints does not rely on pre-defined features, but only requires the recording of web page visits, making our approach actionable. The approach is empirically tested on real-life web logs and compared with matching using regular user-agent features and state-of-the-art embedding techniques. Results in an ecommerce context show sequential features can still obtain strong performance with fewer features, facilitating decision-making on session stitching and inform subsequent related activities such as marketing or customer analysis.

Keywords: session stitching, web analytics, sequence mining, session fingerprinting.

---

1. Introduction

The widespread use of handheld devices offers web users new portals to consult information which combined with a general digitization of services has led to an explosion of web logs leaving traces of people. A major challenge exists in identifying which entries in these web logs, from now on referred to as *sessions*, belong to the same user. The expiration of cookies, the use of different IPs, and sharing devices makes it difficult to link sessions back to the same entity. As a result it is difficult, or even impossible, to derive actionable insights from such high dimensional and seemingly disjoint data to support decision-making as users cannot, e.g., be effectively targeted for marketing campaigns, or offered tailored browsing experiences. Companies have widely invested in web analytics tools such as Google Analytics[1] or Adobe[2] to keep track of users, however, a

[1] `https://analytics.google.com/analytics/web/`

[2] `https://www.adobe.com/uk/analytics/adobe-analytics.html`

dedicated site-based login is often required to deal with the aforementioned problems properly.

Various machine learning and information retrieval approaches were proposed for entity matching, and dedicated approaches exist that focus on web log data specifically such as [1] and [2]. Typically, Google Analytics logs provide features such as the device, browser, geographic location, previously visited web pages, campaigns, and so on. However, the range of variables might vary from implementation to implementation, as well as give rise to a variety of ways to encode particular variables with many possible values leading to high-dimensional feature spaces. Furthermore, user-agent variables (comprising device, browser, and location) are typically very sensitive and cannot be used straightforwardly in case privacy-preserving processing is required. The use of an array of queries with information on gender, age, and location makes the chance of user identification high [3, 4, 5]. In general, query and web visit information is very sensitive and companies often have to dispose of identifiers such as IP [6]. This topic has become especially relevant given General Data Protection Regulation (GDPR) compliance [7] where companies retain information of user visits over international borders. Given the reliance of many session stitching techniques such as the approaches used for benchmarking in the experimental evaluation ([1, 8]) on IP, device, and user-agent information, the proposed approach based on behavioural patterns mined from sequential item sets can be applied from a pure symbolic standpoint which facilitates, e.g., randomization. It has been shown that reverse item set mining is hard [9], which means that sequential item set reversal is at least as hard. While the data used for building the behavioural features is still at scruntiny under GDPR as it is pertaining to users' website browsing behaviour, at least the manual reversal of the features and using them to identify particular individuals will be a close to impossible task. This is evidenced by the fact that the party providing the data for the empirical evaluation used the resulting features and insights over different companies as the risk of privacy leaks were deemed to be non-existent.

To facilitate the interpretability and actionability of session stitching, we

User	Session ID	Timestamp	Web page
User 1	8,401	08:11:20-10/11/2016	/Products/Product1
User 1	8,401	08:12:20-10/11/2016	/Help page/FAQ
User 1	8,401	09:41:32-10/11/2016	/Products/Product1
User 2	8,402	15:10:21-10/11/2016	/Products/Product2
User 3	8,403	04:01:21-10/11/2016	/About/History
User 4	8,404	10:54:23-12/11/2016	/Products/Product1
User 4	8,404	10:55:45-12/11/2016	/Help page/FAQ

Table 1: Example of a web log.

propose an approach founded on fingerprinting the sequence of web page visits (also sometimes referred to as hits) to generate features that can be used to match web log sessions similar to the example present in Table 1. In session stitching, the user label is unavailable and needs to be established by matching different session IDs. For example, in this case user 1 and 4 have similar behavior (visiting the product page of product 1, then visiting the FAQ page) and can be eligible for stitching. The underlying motivation is that users tend to visit either particular sites, e.g., of products of interest, or have a specific order in which they navigate a website, leaving a discernible mark which can be picked up by sequence mining algorithms. More specifically, we use the interesting Behavioral Constraint Miner which excels in generating sequence patterns (also referred to as behavioral constraints) for classification [10] to extract page visit fingerprints. The approach is capable of outperforming traditional sequence mining approaches by returning fewer, though highly discriminating features while retaining a strong level of interpretability which renders them ideal for decision support.

We compare the outcome with two approaches. Firstly, we use typical user-agent information including device, location, and traffic source information. Secondly, network embedding-based approaches are employed, which create a heterogeneous network over the web pages of a website as well as other elements tied to a visit, and their users. Experiments on two real-life datasets show that sequence fingerprinting achieves strong results for sessions stitching that outperforms network embeddings in terms of predictive performance and dimensionality. Also, it is capable to provide interpretable features in constrast

with the latent dimensions provided by embeddings. Although, it does not outperform the fully-fledged information that is available in user-agent information, the advantage of sequence fingerprinting is that it only requires web page visit sequences without any personal information while doing so with only few constraints. This increases interpretability and actionability of insights obtained, as the sequential features used to match users can still be verified by analysts to understand the rationale behind the matching of different session IDs. This is typically hard with traditional approaches using 1,000+ features, or embeddings which cannot be interpreted. Furthermore, it retains a level of generalizability, as the use of fewer features indicates that the matching is not as fine-granular pertaining to intricate combinations of feature values required to match users.

This paper is structured as follows. In the next section, the related work is covered. Section 3 explains the rationale for generating sequential features from sessions, which is used in Section 4 in an empirical evaluation. Finally, Section 6 discusses the implications and threads for future research.

2. Related Work

In this section, an overview of session stitching approaches is given. To this purpose, the field of entity matching is introduced which forms the basis for many session stitching approaches. Next, dedicated sequence-based approaches are covered.

2.1. Entity matching

Session stitching aims at recognising whether two data samples were generated by the same individual or source. Various works in the field of information retrieval address the more general problem of entity matching, also referred to as record linkage, entity resolution, etc. [11]. Given that matching large sets of entities requires a full N-to-N matching, often blocking is used to pre-group entities to avoid computationally intractable calculations. To this purpose, various clustering techniques such as neighbourhood searches can be used [12].

Entities can be matched using various distance functions such as string distance (e.g. edit distance, Jaro distance) and token-based distance metrics (e.g. Jaccard, cosine simlarity) [13, 14], which can serve as input to algorithms such as probabilisitic matching approaches (expectation maximisation) or supervised learning approaches such as Naive Bayes, support vector machines, etc. [15, 16]. Recently, many graph or network-based approaches have surfaced [17] where closure properties can be exploited to recognise similar entities, or to learn other representations of entities such as embeddings [18, 19].

2.2. Entity Matching For Web Data

Tailored entity matching approaches for web logs exist, e.g., [1] investigate a probabilistic approach to stitch together users with probabilistic soft logic from IP, geo location, and user-agent information such as device, operating system, etc. To this purpose, distance metrics are used based on location, as well as textual similarity in IP and user-agent information to train the probability of 2 entities being the same. In [2], various touch points in the web usage process are considered to stitch users together using the Jaccard distance and problem-specific blocking rules for IPs and other identity information. The issue of noisy labels and data that are often making such approaches difficult has been tackled using robust factorization machines in [20]. Location data can also be used towards identifying similar users over multiple devices by using a graph-based approach for user-location relations [21]. Identity graphs [22] can be constructed to provide a probabilistic view on overlapping user behavior in various online sources as well. Per user, a graph is created to connect information of entities to their visiting a website, IPs, logins, etc. These graphs can be matched to obtain a similarity score.

Furthermore, various graph embedding-based approaches have recently surfaced to link entities. node2bits [8] introduces time-aware graph-based embeddings that are learnt over users linked to various elements tied to their web behaviour such as IP, web pages visited, devices used, etc. These embeddings are used for entity linking using supervised learning algorithms. [23] introduce

a similar approach which links entities in a graph over heterogeneous sources such as work interactions (emails) and social events. [24] use graph convolution networks to connect entities from email and other graph-based relations. The latter two works also stress the importance of the privacy-aware processing of data. In the case such sensitive data is needed, pre-clustering has to be applied to (partially) obfuscate any specific personal data.

2.3. Sequence-Based Approaches

While there exists a sub-domain of sequence-to-sequence matching [25], it focuses on aligning similar sequences rather than generating a similarity score. Sequence mining has been applied to web logs before in the form of 2-grams to identify frequent surfing patterns suitable for caching on larger web nodes [26]. [27] use term frequency-inverse document frequency to identify patterns of web page visits which can be used for session identification. [28] introduce sequence rules for entity matching. They assign weights to different attributes tied to entities and match them based on the importance of weights, however, the sequential aspect here is the order of the attributes, rather than ordered item sets. Text-based approaches exist in entity matching where sequential models are applied to documents. [29] use sub-strings similar to the 2-grams approach, and [30] use deep learning models generating embeddings to recognise semantic similarities between text-enriched entity data. In [31], the authors exploit neural networks for identity stitching by increasing misclassification costs for the matches between identical entities which are often sparse. To this purpose, they use the websites visited by users and use word embeddings to vectorise link strings first.

However, the field of sequence mining typically focuses on creating a database of sub-sequences expressed as partial orders/sub-sequences that either summarise most of the sequences, or can re-generate the whole sequence database completely. Sequence mining encompasses a variety of works ranging from text analysis to DNA classification. Most notably, sequence mining has grown from an Apriori-based approach [32] to a range of derivative and novel techniques

such as SPADE [33] and PrefixSPAN [34]. The downside of these approaches is that they tend to generate a high number of partial orders which are not necessarily informative for supervised approaches which require discriminating features. More recently, probabilistic approaches such as Interesting Sequence Miner [35] and MiSeRe [36] have been introduced which focus on generating informative sub-sequences suitable for a concise representation of the general behaviour in a sequence database. In contrast, the interesting Behavioural Constraint Miner (iBCM) [10] was proposed for finding pre-defined sequence patterns (called constraints), defined in linear temporal logic/regular expressions, to classify sequences. These patterns are more expressive than partial orders, and can concisely summarise behaviour in sequences with low dimensionality. Finally, neural network and embedding-based approaches are surfacing for sequence analysis [37]. While they often reach competitive levels of predictive performance, they are limited or unable to provide any insight into the interpretation of underlying important patterns.

In web log analysis, sequence mining has not yet been employed for session stitching. In this study, we investigate the potential of sequence mining in the form of iBCM to obtain concise, interpretable features that can match users from different sessions and subsequently inform decision-making.

2.4. Positioning

The most related studies to this work are [1] and [8] as both works propose an approach specifically tailored towards using web logs in contrast to the other works. As indicated before, the former uses web log data existing of user-agent data (IP, operation system, device type, etc.) and geographic locations. These data are then used in Probabilistic Soft Logic (PSL) rules such as $VisitorIP(V_1, I_1) \wedge VisitorIP(V_2, I_2) \wedge SimIP(I_1, I_2) \Rightarrow SameUser(V_1, V_2)$ where $SimIP$ is a proprietary function to measure the similarity between different IPs which in combination with ground truth on which users have which IPs is used to iteratively estimate the probability for $SameUser$ using PSL inference. Various other experience-based rules are proposed for user-agent and

location variables which are verified over a real-life case sample showing strong results in terms of predictive performance with F1-scores up to 97%. The work is harder to replicate in different settings as distance measures used for locations and IP similarity cannot be applied to every web log. The latter work proposes the use of network embeddings by representing various elements of a web log as a node in a network over time. E.g., IPs are connected to sessions, which are connected to web sites that are visited during that session. By employing typical embeddings approaches based on temporal random walks, temporal contexts are created to avoid overwhelming dimensionality which is subsequently used to hash contexts into binary reprsentations of nodes. Note that other variables such as device, geographic location, etc. can also be incorporated as nodes, although this can quickly increase the size of the network. Experimental evaluation shows strong performance compared to other network embedding approaches on a variety of networks, as well as session stitching. Both approaches are evaluated on proprietary datasets which are not shared for benchmarking or experimental evaluation. Also, ground truth is available in both cases, which is not necessarily the case in web logs. One of the main challenges of session stitching is working in an environment where ground truth might be missing, which is the case for the real-life event log used in the experimental evaluation of this work. The approaches differ in that the former is heavily reliant on domain insights to establish the PSL rule set, while the latter employs a more generic technique which is suitable for web log data and especially time- and sequence-based data. Hence, they represent different takes on the problem of which the proposed sequence fingerprinting technique lies in between.

3. Sequence Fingerprinting of Web Page Visits

Sequence mining algorithms aim to retrieve a set of sequential patterns from a sequence log $\mathcal{L}$. $\sigma = \langle \sigma_1, \sigma_2, ..., \sigma_l \rangle$ is a sequence of length l with the items in the sequence elements of alphabet Σ. Hence a sequence is a string from the language of Σ, $\sigma \in \Sigma^*$. Consider the example in Table 1 where

$\mathcal{L} = \{\langle a, b, a\rangle, \langle c\rangle, \langle d\rangle, \langle a, b\rangle\}$, $\Sigma = \{a, b, c, d\}$ with a = Products/Product1, b = Help page/FAQ, c = Products/Product2, and d = About/History. In our case, Σ comprises the web pages available in the web logs and σ is the string of pages visited during a single session. The goal is to find the smallest set of sub-sequences $\mathcal{F} \in \Sigma^*$ that can replicate $\mathcal{L}$, or that excel in distinguishing sequences for classification. In that case, the presence of each sub-sequence of sequential patterns is used as a feature, i.e., Classifier : $\{0, 1\}^{|\mathcal{F}|} \rightarrow \mathcal{C}$ with $\mathcal{C}$ the classes. Sequence mining algorithms often resort to the concept of support $sup_f = \frac{|\{\sigma | f \in \sigma, \sigma \in \mathcal{L}\}|}{|\mathcal{L}|}$ to avoid eliciting all possible sequences in a database or take a probabilistic approach to generate sequences attributing to differences in sequences. In both cases, sequence patterns are typically increased in length during inference only when the items are frequent enough.

3.1. iBCM

3.1.1. Background

All currently available sequence mining approaches use partial orders to express the sub-sequences returned by their algorithms. iBCM, however, uses behavioral patterns based on the Declare language [38] which have a higher expressive power (i.e. the language they can express) and are capable of capturing various other information, such as whether an item occurs at the beginning of a string, how many times an item occurs, and whether they do not occur after a certain occurrence of another item (negative behavior). An overview of the sequential patterns used can be found in Table 2.

Due to the higher level of expressiveness, iBCM is capable of quickly finding patterns that, when combined, can replicate a sequence quickly without having to discover all underlying partial orders of an increasingly longer length. Besides, these rules are especially useful for classification [10]. iBCM uses support as a parameter to prune items from the alphabet that are infrequent before discovering patterns including these items. Next, it checks the presence of all patterns from Table 2 which include patterns for both one and two items. The latter are checked for all pairs of frequent items given they are present in a

Type	Template	LTL Formula [38]	Regular Expression [39]
Unary	Existence(A,n)	$\Diamond(A \wedge \bigcirc(existence(n-1, A)))$	.*(A.*){n}
	Absence(A,n)	$\neg existence(n, A)$	[^A]*(A?[^A]*){n-1}
	Exactly(A,n)	$existence(n, A) \wedge absence(n+1, A)$	[^A]*(A[^A]*){n}
	Init(A)	A	(A.*)?
	Last(A)	$\Box(A \implies \neg X \neg A)$	.*A
Unordered	Responded existence(A,B)	$\Diamond A \implies \Diamond B$	[^A]*((A.*B.*) |(B.*A.*))?
	Co-existence(A,B)	$\Diamond A \Longleftrightarrow \Diamond B$	[^AB]*((A.*B.*) |(B.*A.*))?
Simple ordered	Response(A,B)	$\Box(A \implies \Diamond B)$	[^A]*(A.*B)*[^A]*
	Precedence(A,B)	$(\neg B \, U \, A) \vee \Box(\neg B)$	[^B]*(A.*B)*[^B]*
	Succession(A,B)	$response(A, B) \wedge precedence(A, B)$	[^AB]*(A.*B)*[^AB]*
Alternating ordered	Alternate response(A,B)	$\Box(A \implies \bigcirc(\neg A \, U \, B))$	[^A]*(A[^A]*B[^A]*)*
	Alternate precedence(A,B)	$precedence(A, B) \wedge \Box(B \implies \bigcirc(precedence(A, B))$	[^B]*(A[^B]*B[^B]*)*
	Alternate succession(A,B)	$altresponse(A, B) \wedge precedence(A, B)$	[^AB]*(A[^AB]*B[^AB]*)*
Chain ordered	Chain response(A,B)	$\Box(A \implies \bigcirc B)$	[^A]*(AB[^A]*)*
	Chain precedence(A,B)	$\Box(\bigcirc B \implies A)$	[^B]*(AB[^B]*)*
	Chain succession(A,B)	$\Box(A \Longleftrightarrow \bigcirc B)$	[^AB]*(AB[^AB]*)*
Negative	Not co-existence(A,B)	$\neg(\Diamond A \wedge \Diamond B)$	[^AB]*((A[^B]*) |(B[^A]*))?
	Not succession(A,B)	$\Box(A \implies \neg(\Diamond B))$	[^A]*(A[^B]*)*
	Not chain succession(A,B)	$\Box(A \implies \neg(\bigcirc B))$	[^A]*(A+[^AB][^A]*)*A*
Choice	Choice(A,B)	$\Diamond A \vee \Diamond B$	.*[AB].*
	Exclusive choice(A,B)	$(\Diamond A \vee \Diamond B) \wedge \neg(\Diamond A \wedge \Diamond B)$	([^B]*A[^B]*) |.*[AB].*([^A]*B[^A]*)

Table 2: An overview of Declare constraint templates with their corresponding LTL formula and regular expression.

sequence. All these checks can be performed using efficient string operations.

3.1.2. Algorithm

Given a minimum support min_sup and an alphabet Σ, iBCM first retains only the most frequent items $\Sigma_f = min\_sup \cdot |\Sigma|$. If $min\_sup = 0.5$ then we obtain $\Sigma_f = \{a, b\}$. Then, for every sequence in $\mathcal{L}$ Algorithm 1 (adopted from [10]) can be applied to retrieve a set of constraints C_σ (referred to as just C in the Algorithm). In line 3, $occ(\sigma_i, \sigma) = \{i \mid \sigma_i = \sigma_j, \forall j \in [1, |\sigma|]\}$, a set which returns all positions of item σ_i in sequence σ, is created to perform the string queries. For example, $occ(a, \sigma_1) = \{1, 3\}$. Lines 5-9 check for unary constraints of single items by checking the absence or presence/existence of an item as well as its cardinality (e.g. existence 2 or 3), and an item's exact position (e.g. at the beginning or end of the sequence:

Algorithm 1 Mining behavioral constraint templates

```
1: procedure mineConstraintsInString(σ, Σ_f)
2:     C ← ∅                                                    ▷ C is a set of constraints outputted
3:     for σ_i ∈ σ do occ(σ_i, σ) ← i
4:     for a ∈ Σ_f ∩ Σ_σ do                                     ▷ Σ_σ is the alphabet of the sequence
5:         if |occ(a, σ)| = 0 then C ← C ∪ absence(a, 1)        ▷ Unary constraints
6:         else if |occ(a, σ)| > 2 then C ← C ∪ existence(a, 3)
7:         else C ← C ∪ exactly(a, |occ(a, σ)|)
8:         if 1 ∈ occ(a, σ) then C ← C ∪ init(a)
9:         if |σ| ∈ occ(a, σ) then C ← C ∪ last(a)
10:        for b ∈ {Σ_f ∩ Σ_σ} \ {a} do                         ▷ Binary constraints
11:            C ← C ∪ CoExist(a, b)
12:            if min(occ(a, σ)) < min(occ(b, σ)) then
13:                C ← C ∪ prec(a, b)
14:                i ← min(occ(b, σ))
15:                chain ← (i − 1) ∈ occ(a, σ), continue ← ⊤
16:                while ∃n ∈ occ(b, σ), n > i ∧ continue do
17:                    if ∃p ∈ occ(a, σ), i < p < n then i ← n
18:                        if ¬chain ∨ (n − 1) ∉ occ(a, σ) then chain ← ¬
19:                    else continue ← ¬
20:                if continue ∧ |occ(b, σ)| > 1 then C ← C ∪ altPrec(a, b)
21:                if chain then C ← C ∪ chainPrec(a, b)
22:            if max(occ(a, σ)) < max(occ(b, σ)) then
23:                C ← C ∪ resp(a, b)
24:                if max(occ(a, σ)) < min(occ(b, σ)) then
25:                    C ← C ∪ notSuc(a, b)
26:                i ← min(occ(a, σ))
27:                chain ← (i + 1) ∈ occ(b, σ), continue ← ⊤
28:                while ∃n ∈ occ(a, σ), n > i ∧ continue do
29:                    if ∃p ∈ occ(b, σ), i < p < n then
30:                        i ← n
31:                        if ¬chain ∨ (n + 1) ∉ occ(b, σ) then chain ← ¬
32:                    else continue ← ¬
33:                if continue ∧ |occ(a, σ)| > 1 then C ← C ∪ altResp(a, b)
34:                if chain then C ← C ∪ chainResp(a, b)
35:            add succession if (alternate/chain) response and precedence
36:            if b ∉ Σ_σ ∧ b ∈ A then C ← C ∪ exclChoi(a, b)
37:    return C
```

init/last). Lines 10-36 check the combination of two-item sets to establish co-existence (occurring together regardless of their order), precedence (1 item appears before), alternation (items occur in alternating order), and chain-based behaviour (items occur right after each other in a sequence). Also negative constraints, such as exclusive choice between two items, or non succession are discovered. After running the Algorithm, we obtain a feature set, e.g., $C_{\sigma_1} = \{\text{exactly}(a, 2), \text{exactly}(b, 1), \text{init}(a), \text{last}(a), \text{chain precedence}(a, b),$ not succession$(a, b)\}$. Note that a hierarchical reduction is applied to the constraints afterwards where chain/alternate precedence/response/succession ranks

higher than (alternate) precedence/response/succession [40] to retain the most expressive constraints. Finally, the algorithm returns $\mathcal{F} = \bigcap_{\sigma \in \mathcal{L}} C_\sigma$, the full set of constraints which serves as the features used in the subsequent classification. Each sequence in $\mathcal{L}$ can then be labelled according to whether the constraints hold for that sequence. This sequence fingerprint can be used to compare two sequences and fed to a classifier, e.g., the end result of matching sessions with (a subset of the features) is captured in Table 3.

	sequence 1				**sequence 2**			
	exactly(a,2)	**init(a)**	**last(a)**	**not succession(a,b)**	**exactly(a,2)**	**init(a)**	**last(a)**	**not succession(a,b)**
$\langle a,b,a\rangle$,$\langle c\rangle$	1	1	1	0	0	0	0	0
$\langle a,b,a\rangle$,$\langle d\rangle$	1	1	1	0	0	0	0	0
$\langle a,b,a\rangle$,$\langle a,b\rangle$	1	1	1	0	0	1	0	0

Table 3: An example of a session sequence matched with 3 other session sequences using sequence constraints.

3.1.3. Performance

As can be seen from Algorithm 1, three major factors have an impact on the number of features generated and runtime. Firstly, the minimum support will affect the size of Σ_f and hence the number of pairs of items that have to be checked for binary constraints. This means that the impact of the separate sequences is strong as well. The alphabet of each sequence can further reduce the number of pairs that have to be checked. Hence, the dispersion of the alphabet in combination with the minimum support both influence the number of constraints returned. Finally, iBCM is efficient because most constraints can be checked without traversing the sequence and hence are just discovered using boolean checks. Nevertheless, the checks for alternate and chain constraints (lines 16-21 and lines 27-34) can become expensive for long strings. However, given that these constraints are very specific, they typically do not hold for many sequences which makes issue often non-existent.

3.2. Considerations For Web Log Analysis

A web page sequence fingerprinting approach can be performed by any sequence mining approach. However, iBCM outperforms other approaches for classification in terms of accuracy, number of constraints necessary, and execution time [10], even LSTM-based approaches. Word embeddings such as word2vec [41], which are often used for text mining, are not appropriate in this scenario given the relatively short length of web page sequences.

The benefit of generating sequential features is their interpretability. Instead of using LSTMs or embedding-based approaches which are essentially black boxes, sequential features can still be readily interpreted. Especially in matching, it is possible to gauge what matching variables are particularly different for matching and non-matching sessions, or drive predictive models.

There are a few considerations to make that are specific to web log data. In web sites, fingerprinting can be troublesome in case there is a large alphabet because of the size of the website. To overcome this issue, we consider links at various levels of depth, e.g., in Table 1 only the first part of the link can be considered such as `Products`, `Help page`, and `About`, or longer lengths including the full page string. Besides, users can generate very long page visit strings which might hamper overall performance as discussed in Section 3.1.3. In the experimental analysis, the number of constraints generated will give the best idea of how computationally expensive the generation of sequence fingerprints is.

4. Empirical Evaluation

In this section, we apply sequence fingerprinting to two Google Analytics data logs for empirical evaluation. To this purpose, we compare this with two other methods. Firstly, we use user-agent data which contains the most individual-specific information on device, location, and so on. Secondly, we use node2bits [8] as it is one of the most recent embedding-based approaches which

focus on web log data. Finally, iBCM is used to generate sequential features to match sessions.

4.1. Data

The data was provided by a Digital Marketing and SEO company and spans 1 month's worth of Google Analytics data of a significant retailer and a leisure goods retailer, referred to as case 1 and 2 respectively. For the former, 1,642,984 sessions were retrieved, for the latter 744,984. Statistics on the datasets can be found in Table 4. The parameters influencing the size of training and test set are discussed below. Sessions with the same visitor ID are used as matches (returning visitors), with a detailed sampling approach being discussed below. Note that for dataset 2, the proportion of returning visitors in the dataset is higher.

		dataset 1				
mpl	**mv**	**Train**	**Test**	**Ret. Train**	**Ret. Test**	**Ret. All**
2	2	294,369	145,745	7,489	6,778	14,267
	3	294,369	145,745	539	792	1,331
	5	294,369	145,745	19	34	53
3	2	226,777	112,507	5,295	4,930	10,225
	3	226,777	112,507	363	551	914
	5	226,777	112,507	7	24	31
		dataset 2				
2	2	104,880	52,450	9,792	8,813	18,605
	3	104,880	52,450	1,862	2,607	4,469
	5	104,880	52,450	134	377	511
3	2	89,126	44,415	7,413	6,822	14,235
	3	89,126	44,415	1,246	1,911	3,157
	5	89,126	44,415	71	222	293

Table 4: Overview of the datasets' size given the different parameter configurations. Train/test indicate the size of the training and test set respectively. Ret. train/ret. test/ret. all stand for the number of returning visitors for the training/test/whole dataset respectively.

4.2. Approaches and Sampling

As discussed in Section 2, there are numerous branches in entity matching, in particular for session matching. Some techniques are better applicable than

others. For example, using probabilistic soft logic [1] relies on an extensive example base to iteratively generate inference rules to obtain probabilities. Given that no other blocking ID (such as IP) was available, there are too few examples to obtain any meaningful inferences. Therefore, we use a general feature-based approach which employs similar variables (these codes are used in the result tables below):

- Device or browser version (1)
- Geographic location (2)
- Traffic source (3): the origin of traffic (e.g. Google.com)
- Traffic source medium (3): the source's category (e.g. organic)
- Session start time: hour/day (4)

This evaluation approach is similar to the classifier evaluation of [1] which boasts results close or similar to PSL. For a more detailed description of these variables and how they are recorded we refer to the Google Analytics definitions[3].

We will use these variables on/off with a standard classifier to gauge their importance in matching sessions. Besides, we use the embedding-based approach node2bits [8] which is tailored to using heterogeneous graphs representing sessions and web page visits. The graph is constructed by connecting sessions with page visits as they occur, which in combination with the underlying word2vec-based embedding constitutes a sequence-based result in the form of a $|\mathcal{F}|$-size vector per node. Besides, nodes for device (1) and location (2) are included as well to see whether these improve the embeddings. Note that the approach can also incorporate timing information, however, this resulted in empty embeddings during the experiments. Hence, only order is considered which makes for a good benchmark for the timing-agnostic approach of iBCM. node2bits

[3] `https://developers.google.com/analytics/devguides/collection/analyticsjs/field-reference`

was originally verified using artificially-generated nodes based on existing data points.

In this paper, we adhere to the following sampling strategies given a full-on match of all data points is intractable. For every session matched based on visitor ID in the training set, 5 other randomly-selected sessions are used as non-matches to obtain negative examples. In the test set, we again match all newly-occurring visitor IDs' sessions with 5 other sessions for verification. A similar approach was used in [1] and [31] by retaining a particular ratio of matching/non-matching sessions in the training and test sets. Note that in a common real-life setup, previous IDs would be included in the matching in a second (test round), however, the sampling setup chosen here prevents any data leaking in from the training set.

4.3. Experimental Setup

The features $\mathcal{F}$ generated by each approach per session s_i are used to create a feature vector $f_{s_i} \in \{0,1\}^{\mathcal{F}}$. Next, every session s_i with an ID matching other sessions are considered, and 5 counter examples are generated to reflect the imbalance caused by the low level of matching sessions. Increasing the number would be more realistic, however, would lead to unnecessarily large datasets for experimental purposes. Thus, we obtain a final feature vector for every match, $f_m = f_{s_1}, f_{s_2}$, the concatenation of feature vectors of both sessions. For the general user-agent feature-based approach, all variables are one-hot-encoded. Finally, these vectors are passed on to a classification algorithm with their respective label indicating whether they match or not (have the same user/visitor ID). After testing various classifiers, i.e., logistic regression, naive Bayes, and support vector machines, we only report the results of random forests given their superior classification results. As two standard binary classification metrics we use the F1-score (the harmonic mean of recall and precision) and the area under receiver operating characteristic curve (AUC) for both are capable of adequately measuring imbalanced datasets. This is reflected in the fact that increasing or decreasing the number of counter examples did not affect AUC

drastically.

The iBCM implementation was written in Java and can be found here[4]. The evaluation was run multi-threaded on a Xeon E3-1230 v5 CPU with 32GB of memory which makes it possible to retrieve all behavioural patterns in less than 30 seconds for the first dataset, and les than 2 minutes for the second dataset which is denser (see Table 5).

4.4. Parameters

The evaluation considers various parameters that have an impact on the outcome and usability of the different approaches:

- Minimum number of visits (*mv* - values used: 2,3): number of sessions/visits before a visitor with the same ID is considered for matching. Given the low numbers for retained sessions at $mv = 5$ as reported in Table 4, the final results are not included.

- Minimum page visit length (*mpl* - values used: 2,3,5): web logs often suffer form high bounce rates. We set the minimum web page sequence at various lengths to reflect how many pages a user needs to have visited before being considered as a sample.

- Page string cutoff (*pc* - values used: 2,3): as discussed in Section 3, the alphabet of web pages can have a significant impact on the number of sequential features generated, and hence on the dimensionality of the matching. Various cut-offs will be considered to see whether deeper web page information results in more informative features. This parameter is only relevant to iBCM/node2bits. Besides, note that at cutoff 2 we obtain the first page level (e.g. products, info, help, etc.), as the first is reserved for the top-level domain (the website itself).

- Support level (*sup.* - values used: 0.1, 0.01, 0.05): iBCM uses support to

[4] `https://github.com/JohannesDeSmedt/iBCM`

only consider web pages that are present frequently enough which controls how many features are generated eventually.

- Use of device (1)/location (2)/traffic source/medium (3)/timing (4): the inclusion of these various variable types are considered for the feature-based approach and node2bits.

An overview of the influence of parameters on the number of samples, number of samples with enough sequences, and the number of returners in the training/test/full dataset can be found in Table 4. It is clear that the minimum page length mpl requirement cuts a significant proportion of the overall number of sessions and that the minimum visit number drastically reduces the number of eligible IDs used for matching. It shows that in both datasets, very few IDs are returning more than 3 times although this proportion is higher for dataset 2. Also, more sessions are retained for $mpl = 3$, meaning page visit lengths are longer than for dataset 1. This might be related to the type of online business, or the fact that cookies expire sooner making the visitor ID less reliable. The table also illustrates that the proportion of sessions stemming from returning visitors is low with a ratio of 1-3%. This makes the matching problem very unbalanced given the discrepancy between the few sessions that actual match.

An overview of the size of the web page alphabet size Σ of the training set can be found in Table 5. It appears that the web site hierarchies are different in both cases. For dataset 1, there is a bigger difference when having $pc = 2$ compared to $pc = 3$, meaning there are fewer top-level pages but very detailed links when deeper cutoffs are considered. For dataset 2, the difference is less pronounced and the top level covers a very wide range of pages already. This indicates that the web site has a much flatter hierarchy. Nevertheless, the alphabet size of dataset 2 is considerably higher than of dataset 1. The ratio between page-cutoff values for a longer minimum page length is similar, meaning that the top level pages of the datasets are well-visited regardless, and that the longer web page visit sequences often contain similar top-level and deeper-level web pages as shorter visits.

		alphabet size	
mpl	**pc**	**dataset 1**	**dataset 2**
2	2	1,211	92,428
2	3	77,184	130,500
3	2	1,160	87,840
3	3	69,994	125,592

Table 5: Overview of the alphabet size of the training set for both datasets depending on minimum page length and page-cutoff.

4.5. Results

In this section, both the analysis of the predictive results, as well as the importance of mined sequential features are covered.

4.5.1. Matching

The results of applying the three approaches can be found in Tables 6 to 11.

Overall, the AUC and F1-score of the user-agent features is consistently the highest of all the approaches. When using all features, it achieves AUC up to 98/98, 96/97% for the two values of mv and datasets 1/2, respectively. The F1-sore is lower with 93/92, 92/91%, which is mostly due to precision being lower. The size of the feature space is comparable between both datasets, indicating that there are a similar number of values for device, location, etc. The incorporation of the device information (1) and location information (2) boosts AUC and F1-score the most, while using just the medium/channel information (3) achieves reasonable results still. Using timing information (4) results in a weak AUC and poor F1-score for either dataset possibly due to the low dimensionality and little variance introduced by the timing features. The minimum page length mpl has no significant impact, as it only changes the sample slightly as demonstrated in Table 4.

Using iBCM results in an AUC with peaks of 80/72, 81/68% for the two values of mv and datasets 1/2, which is significantly lower. Also, the F1-score is lower with 80/74, 71/73%, with these results not necessarily being coupled to the best AUC in terms of parameter usage. The number of behavioural features generated, despite the bigger alphabet size for dataset 2, is similar to

				mv=2				mv=3			
				mpl=2		mpl=3		mpl=2		mpl=3	
pc	$\|F\|$	**(1)**	**(2)**	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**
2	64	x	x	0.489	0.601	0.459	0.593	0.379	0.596	0.415	0.589
		x		0.552	0.610	0.546	0.607	0.460	0.617	0.483	0.612
			x	0.576	**0.614**	0.558	**0.608**	0.495	0.624	0.594	0.609
				0.697	0.600	**0.673**	0.596	**0.754**	0.599	**0.701**	0.593
	128	x	x	0.453	0.593	0.435	0.589	0.420	0.589	0.424	0.583
		x		0.548	0.607	0.526	0.607	0.456	0.618	0.530	0.608
			x	0.566	0.612	0.538	0.606	0.513	**0.626**	0.608	**0.611**
				0.682	0.600	0.670	0.594	0.722	0.593	0.674	0.587
	256	x	x	0.443	0.584	0.436	0.587	0.362	0.584	0.319	0.585
		x		0.549	0.609	0.553	0.605	0.478	0.615	0.520	0.602
			x	0.561	0.612	0.568	**0.608**	0.474	0.622	0.509	0.601
				0.691	0.600	0.637	0.593	0.647	0.597	0.631	0.587
3	64	x	x	0.360	0.563	0.359	0.572	0.377	0.564	0.408	0.568
		x		0.434	0.598	0.427	0.595	0.458	0.599	0.435	0.596
			x	0.438	0.607	0.411	0.605	0.458	0.607	0.458	0.605
				0.497	0.652	0.445	0.643	0.503	0.647	0.474	0.647
	128	x	x	0.485	0.611	0.474	0.614	0.541	0.616	0.524	0.635
		x		0.521	0.635	0.513	0.637	0.554	0.649	0.498	0.653
			x	0.520	0.636	0.515	0.643	0.497	0.641	0.553	0.652
				0.498	0.650	0.461	0.649	0.543	0.662	0.532	0.668
	256	x	x	0.549	0.639	**0.562**	0.656	**0.586**	0.643	0.568	0.671
		x		**0.566**	0.650	0.541	0.653	0.582	0.656	0.535	0.662
			x	0.554	0.646	0.526	0.656	0.564	0.651	**0.579**	0.665
				0.511	**0.659**	0.501	**0.661**	0.527	**0.673**	0.549	**0.688**

Table 6: An overview of the results for dataset 1 using iBCM. The gray scale for AUC indicates relatively performance over all parameter settings.

dataset 1 for higher support values (0.1/0.05), and lower for lower support values (0.01) indicating that these pages from the large alphabet are not as frequently occurring as the less diverse page alphabet of dataset 1. The minimum page length *mpl* does not impact AUC and F1-score performance drastically. The page-cutoff *pc* has an effect on AUC with an overall lower score for dataset 2 when $pc = 3$. Using more detailed, low-level page information leading to more behavioural patterns to be discovered hence does not necessarily result in better performance, but most of the best results for dataset 1 can be found for a longer page-cutoff in combination with lower support. The F1-score mimics this result as well, with a higher page-cutoff resulting in the best performance due to a higher precision. The lower the support, and the higher the number of features returned, the better the results for dataset 1 in terms of AUC/F1, although the

		mv=2						mv=3					
		mpl=2			mpl=3			mpl=2			mpl=3		
pc	**sup.**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**
2	0.1	10	0.707	0.767	9	0.689	0.779	10	0.649	0.785	9	0.670	0.793
2	0.05	14	0.694	0.768	13	0.662	0.778	14	0.612	0.783	13	0.557	0.796
2	0.01	27	0.695	0.770	25	0.671	0.778	27	0.668	0.784	25	0.703	0.799
3	0.1	14	**0.710**	0.769	18	0.667	0.773	14	0.690	0.766	18	0.670	0.772
3	0.05	45	0.668	0.765	60	0.662	0.782	45	0.644	0.777	60	0.692	0.807
3	0.01	249	0.678	**0.774**	311	**0.702**	**0.796**	249	**0.711**	**0.793**	311	0.710	**0.813**

Table 7: An overview of the results for dataset 1 using node2bits.

				mv=2						mv=3					
				mpl=2			mpl=3			mpl=2			mpl=3		
(1)	**(2)**	**(3)**	**(4)**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**
x	x	x	x	3,422	0.857	0.940	2,902	0.809	0.926	3,422	0.848	0.898	2,902	0.858	0.891
x	x		x	2,211	0.827	0.958	1,879	0.852	0.962	2,211	0.804	0.908	1,879	0.851	0.919
x	x	x		3,419	0.909	0.965	2,899	0.913	0.965	3,419	0.900	0.950	2,899	0.890	0.951
x	x			2,208	**0.933**	**0.977**	1,876	**0.927**	**0.975**	2,208	**0.915**	**0.962**	1,876	**0.913**	**0.954**
x		x	x	3,261	0.735	0.902	2,757	0.761	0.909	3,261	0.739	0.860	2,757	0.770	0.868
x			x	2,050	0.721	0.923	1,734	0.697	0.907	2,050	0.676	0.881	1,734	0.730	0.884
x		x		3,258	0.875	0.951	2,754	0.864	0.949	3,258	0.835	0.924	2,754	0.830	0.918
x				2,047	0.872	0.959	1,731	0.854	0.944	2,047	0.850	0.945	1,731	0.840	0.942
	x	x	x	1,375	0.711	0.866	1,171	0.698	0.858	1,375	0.749	0.838	1,171	0.622	0.810
	x		x	164	0.704	0.874	148	0.721	0.875	164	0.609	0.831	148	0.697	0.839
	x	x		1,372	0.839	0.912	1,168	0.834	0.910	1,372	0.827	0.895	1,168	0.819	0.885
	x			161	0.815	0.909	145	0.811	0.908	161	0.807	0.904	145	0.798	0.893
		x	x	1,214	0.527	0.723	1,026	0.493	0.703	1,214	0.564	0.724	1,026	0.535	0.711
			x	3	0.337	0.587	3	0.328	0.564	3	0.306	0.553	3	0.303	0.545
		x		1,211	0.805	0.768	1,023	0.802	0.771	1,211	0.779	0.771	1,023	0.791	0.779

Table 8: An overview of the results for dataset 1 using the user-agent feature-based approach.

increase is not strong meaning that even for lower support values good AUC/F1 can be obtained with fewer features. For dataset 2, the effects of support are not as clear, with the best AUC/F1 typically with lower support and lower page-cutoff. In general it does not seem to pay off to use very low support and longer page-cutoff, especially when the alphabet of the web pages is lower which is the case for dataset 1.

Finally, node2bits achieves the lowest AUC, with peaks at 66/67 and 69/63% for the two values of mv and datasets 1/2, which is the lowest of the three approaches. For the F1-score, this comes down to 70/64 and 75/60%. The results

				mv=2				mv=3			
				mpl=2		mpl=3		mpl=2		mpl=3	
pc	$\|F\|$	**(1)**	**(2)**	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**
2	64	x	x	0.373	0.593	0.383	0.587	0.361	0.571	0.364	0.571
		x		0.400	0.606	0.397	0.594	0.376	0.581	0.371	0.574
			x	0.404	0.610	0.384	0.601	0.373	0.586	0.390	0.580
				0.473	**0.673**	0.367	**0.647**	**0.473**	**0.634**	0.405	**0.617**
	128	x	x	0.447	0.599	0.440	0.594	0.425	0.582	0.428	0.579
		x		0.432	0.613	0.407	0.603	0.418	0.594	0.402	0.584
			x	0.422	0.621	0.412	0.606	0.390	0.594	0.401	0.580
				0.442	0.662	0.415	0.636	0.444	0.627	0.412	0.613
	256	x	x	**0.480**	0.610	**0.480**	0.608	0.453	0.593	**0.472**	0.588
		x		0.454	0.622	0.447	0.611	0.436	0.597	0.441	0.598
			x	0.434	0.626	0.438	0.613	0.441	0.605	0.437	0.596
				0.434	0.660	0.423	0.637	0.423	0.624	0.446	0.612
3	64	x	x	0.401	0.586	0.359	0.566	0.376	0.560	0.356	0.549
		x		0.411	0.593	0.381	0.580	0.393	0.570	0.372	0.570
			x	0.436	0.602	0.384	0.589	0.401	0.579	0.369	0.573
				0.638	**0.667**	**0.441**	**0.635**	**0.595**	**0.630**	0.426	**0.610**
	128	x	x	0.441	0.590	0.409	0.575	0.412	0.568	0.386	0.558
		x		0.443	0.600	0.411	0.583	0.430	0.576	0.395	0.568
			x	0.446	0.609	0.404	0.590	0.419	0.579	0.420	0.572
				0.508	0.645	0.414	0.616	0.506	0.616	0.408	0.601
	256	x	x	0.486	0.603	0.439	0.590	0.469	0.579	**0.436**	0.571
		x		0.452	0.606	0.431	0.595	0.443	0.586	0.429	0.584
			x	0.467	0.612	0.415	0.592	0.447	0.587	0.431	0.579
				0.509	0.643	0.417	0.614	0.496	0.620	0.424	0.603

Table 9: An overview of the results for dataset 2 using iBCM.

are relatively insensitive to the embedding dimension ($|\mathcal{F}|$), and are highest when neither device, nor location are used which is especially pronounced for dataset 2 in terms of AUC and dataset 1 in terms of F1-score. In general, the higher embedding dimensions do contribute to better AUC for dataset 1, but it is unclear whether the incorporation of the other variables is necessary. The higher page-cutoff does lead to better results for dataset 1 in terms of AUC, but not F1-score. The impact of incorporating device information (1) does not seem to have any impact, while incorporating location information (2) slightly boosts performance. Nevertheless, it seems these results are not as effective in terms of predictive performance as the two other approaches.

In general, retrieving page information using *pc* at a deeper level does not necessarily result in better performance for the two sequence-based approaches. Given $mv = 2/3$, it appears that mostly node2bits can leverage some extra

			mv=2						mv=3					
			mpl=2			mpl=3			mpl=2			mpl=3		
pc	**mv**	**sup.**	**—F—**	**F1**	**AUC**	**—F—**	**F1**	**AUC**	**—F—**	**F1**	**AUC**	**—F—**	**F1**	**AUC**
2	2	0.1	10	0.733	0.692	10	0.711	0.684	10	0.700	0.666	10	**0.710**	0.664
2	2	0.05	20	0.729	0.697	22	0.712	0.700	20	0.694	0.670	22	0.689	**0.678**
2	2	0.01	103	0.708	**0.715**	119	0.675	**0.707**	103	0.675	**0.684**	119	0.628	0.673
3	2	0.1	10	**0.742**	0.638	12	**0.718**	0.628	10	**0.729**	0.629	12	0.686	0.626
3	2	0.05	20	0.741	0.647	23	**0.718**	0.656	20	0.691	0.634	23	0.691	0.649
3	2	0.01	164	0.707	0.690	197	0.674	0.678	164	0.680	0.670	197	0.645	0.661

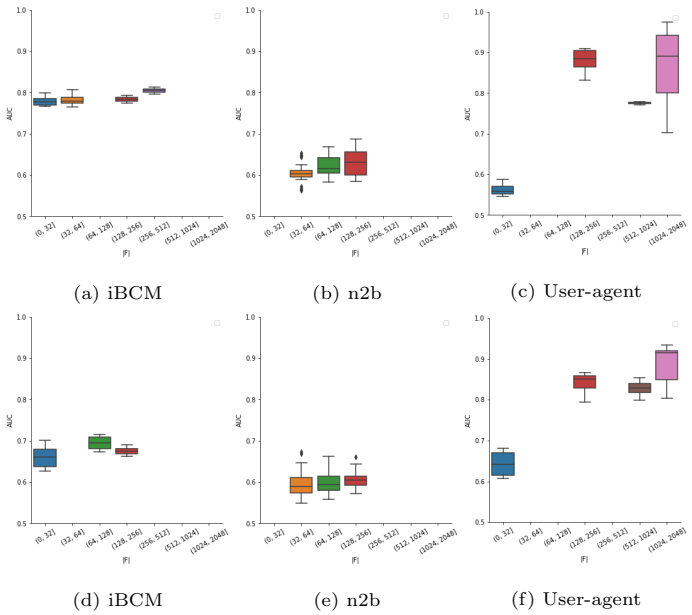
Table 10: An overview of the results for dataset 2 using node2bits.

				mv=2						mv=3					
				mpl=2			mpl=3			mpl=2			mpl=3		
(1)	**(2)**	**(3)**	**(4)**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**	$\|F\|$	**F1**	**AUC**
x	x	x	x	3,559	0.918	0.970	3,331	**0.915**	0.967	3,559	0.909	0.965	3,331	0.901	0.958
x	x		x	2,492	0.887	0.963	2,357	0.878	0.958	2,492	0.859	0.952	2,357	0.847	0.943
x	x	x		3,556	**0.922**	0.973	3,328	0.913	0.968	3,556	**0.910**	**0.965**	3,328	**0.911**	**0.963**
x	x			2,489	0.874	**0.982**	2,354	0.882	**0.971**	2,489	0.879	0.937	2,354	0.869	0.919
x		x	x	3,368	0.864	0.949	3,142	0.866	0.946	3,368	0.853	0.941	3,142	0.851	0.931
x			x	2,301	0.791	0.923	2,168	0.785	0.918	2,301	0.761	0.908	2,168	0.748	0.892
x		x		3,365	0.872	0.958	3,139	0.869	0.958	3,365	0.852	0.947	3,139	0.852	0.944
x				2,298	0.861	0.957	2,165	0.866	0.941	2,298	0.851	0.909	2,165	0.844	0.892
	x	x	x	1,261	0.828	0.920	1,166	0.823	0.917	1,261	0.825	0.915	1,166	0.805	0.904
	x		x	194	0.766	0.853	192	0.739	0.834	194	0.690	0.814	192	0.654	0.794
	x	x		1,258	0.856	0.934	1,163	0.864	0.931	1,258	0.842	0.922	1,163	0.848	0.917
	x			191	0.873	0.867	189	0.869	0.858	191	0.867	0.864	189	0.807	0.847
		x	x	1,070	0.711	0.824	977	0.707	0.823	1,070	0.678	0.803	977	0.674	0.800
			x	3	0.728	0.681	3	0.656	0.667	3	0.499	0.617	3	0.460	0.607
		x		1,067	0.825	0.854	974	0.830	0.854	1,067	0.819	0.833	974	0.819	0.835

Table 11: An overview of the results for dataset 2 using the user-agent feature-based approach.

performance with a higher *pc*, while for iBCM mostly the F1-score increases while AUC is lower. This means that the extra features generated by iBCM are likely not frequent enough to be discriminative to match even very specific visitors.

Overall, there seems to be a prevalent trade-off between the number of features and predictive performance as illustrated in Figure 1. The best precision and AUC are obtained by using user-agent features which often have a high number of values resulting in feature space sizes exceeding 2,000, with the lowest value achieving AUC over 70% being 145 for dataset 1 and 191 for dataset 2. iBCM reports similar AUC regardless of support values and hence feature



(a) iBCM (b) n2b (c) User-agent

(d) iBCM (e) n2b (f) User-agent

Figure 1: Boxplots of the AUC results for different bins for the number of features of all approaches (top row = dataset 1, bottom row = dataset 2).

space size. Lower support values resulting in $|\mathcal{F}|$ as low as 10 can already obtain AUC close to the best performance. For example, iBCM achieves $|\mathcal{F}|$ 9/20, and AUC 80/70% for datasets 1/2 where user-agent features achieve $|\mathcal{F}|$ 145/189, and AUC 91/86%. This indicates that obtaining a very detailed picture is still necessary to uniquely identify users as the same, however, using sequence fingerprinting can already provide an anonymous proxy which is capable of achieving good results. While it is possible to reduce the feature space of user-agent features using, e.g., PCA, this would make the interpretation of results harder again. Using wrapper methods to select features on the other hand is computationally costly.

4.5.2. Variable Analysis and Decision Making

As previously discussed, using sequential features has the main benefit of offering results that are still interpretable instead of being part of an embedded structure or neural network. Hence, it is possible to grasp how each of them

influences the predictive outcome.

In Table 12, an overview is given of the behavioural features that had the highest reduction in Gini impurity throughout the random forest that generated the highest AUC in Tables 6 and 11 (dataset 1, AUC 81% for support 0.01 with a minimum number of visits of 2, page cutoff of 3, and minimum page length 3). In this case, fewer two-item constraint were withheld, and mostly cardinality (absence/existence/exactly) seems to influence the prediction. Note that 'direct' indicates the channel used for entering a website.

Some clear indications can be gathered from these features on how matching was performed. It appears that the absence (yes/no) of particular product categories are discriminatory (clothing and groceries), and most importantly, the absence of a checkout/basket/order step is effective in finding matching sessions. This indicates that returning visitors are likely to be the ones who eventually make a purchase and are more identifiable towards having similar behaviour. Furthermore, the consultation of the store-locator page and accessing a wishlist are strong predictors, again hinting that visitors further on in the conversion process are easier to stitch. The constraints containing less frequent items which are not reducing Gini impurity as much on a global level act as further match steering for individuals in combination with these major trends. Binary constraints also indicate that particular sequences are predictive, e.g., visiting clothing accessories after search results (response), or not returning to the more general clothing/groceries page after consulting a particular accessory or product respectively (not succession). In combination, all these features are capable of discerning which users are similar. E.g., it might be that particular customers are prone to visit some websites twice before visiting a wishlist and checkout, while others proceed to checkout immediately. Added with product information (e.g., did the customer buy groceries or clothing), this can quickly create an intricate, unique picture of a visitor. This can aid in verifying whether the matching makes sense, e.g., that two customers are not matched with completely different product interests, or different browsing behaviour. This would be an arduous task for, e.g., matching based on IP address or a combination of

Constraint	Antecedent	Consequent
Absence	'clubcard', 'boost'	
Absence	'direct', 'clothing-accessories'	
Absence	'direct', 'clothing'	
Absence	'direct', 'my wishlist'	
Absence	'direct', 'search-results'	
Absence	'groceries'	
Absence	'groceries', 'basket'	
Absence	'groceries', 'checkout'	
Absence	'groceries', 'order'	
Absence	'groceries', 'product'	
Absence	'store-locator', 'uk'	
Chain response	'groceries'	'groceries', 'product'
Exactly	'store-locator', 'uk'	
Exactly(2)	'direct', 'search-results'	
Exactly(2)	'groceries'	
Exactly(2)	'groceries', 'product'	
Existence(3)	'direct', 'clothing-accessories'	
Existence(3)	'direct', 'my wishlist'	
Existence(3)	'direct', 'search-results'	
Existence(3)	'groceries', 'checkout'	
Existence(3)	'groceries', 'product'	
Init	'direct', 'clothing-accessories'	
Init	'groceries'	
Init	'groceries', 'product'	
Init	'store-locator', 'uk'	
Not succession	'direct', 'clothing-accessories'	'direct', 'clothing'
Not succession	'groceries', 'product'	'groceries'
Precedence	'groceries', 'basket'	'groceries', 'product'
Response	'direct', 'search-results'	'direct', 'clothing-accessories'
Succession	'groceries'	'groceries', 'product'

Table 12: Overview of the most frequently similar sequence patterns for matching sessions.

geolocation and device where often values are too specific to be matched without extensive prior knowledge. Finally, it also gives insights into users' behavior in general, and how they navigate a website in different ways which can support web site layout optimisation and marketing efforts.

5. Managerial implications

In this paper we elaborate on the web page visit fingerprinting approach, which addresses the challenges marketing managers are facing. Specifically, this

paper derives the following managerial implications:

- First, currently web users access ecommerce websites using different devices, or they access the sites using a device they share with others. This is problematic for marketing managers, as multi-device and shared device use renders them unable to identify individual users and depict their web browsing patterns. As a result, they are unable to design websites that would suit users' individual needs and behaviours. This paper shows that session stitching can be performed using sequential rules from website data which allows to identify users based on browsing paths, product discovery, and more to create an intricate image of how a website is used, as well as the profiles of the people visiting it. Marketing managers can use this approach to develop a profile of online persona that will depict a user who uses multiple devices to access the site, or shares a device with others. Such profile will allow them to gain a good understanding of individual web users browsing patterns, which can inform web design. This is critical, as web design does not only effect consumer online experience but also conversion rates [42].

- Key to the development of a profile of online persona is data. Although it may seem that a vast array of data is available as a result of users' engagement with the website, access to such data often depends on particular vendors such as Google Analytics. Moreover, from the practical point of view, not all available information can be easily used and processed. To mitigate those issues, marketing managers can rely on our approach as the use of sequence fingerprinting is not dependent of particular vendors, and it can be used across platforms. Moreover, since our approach does not rely on sensitive data types such as IP and location, or require domain knowledge to match such variables as introduced in [1], managers can easily process available information and derive insights. In practice, this has been instrumental to the company providing the data as pre-processing geolocation data is often considered tedious. It is also vital to other ecom-

merce companies, as although they have access to the 'new gold' struggle to use it for decision-making.

- One of the fundamental tasks of marketing managers is to ensure users privacy online. GDPR provides a useful guideline to follow, but at the same time it limits marketing managers in terms of information they can use. Our approach minimizes limitations imposed by GDPR and allows to extract useful information without compromising user privacy as item sets are hard to reverse engineer [9], meaning despite the use of user-based information a level of privacy can be retained. This allowed the company providing the data to also use the insights with different ecommerce clients by having initial customer profiles ready. Other companies can also use this approach to enrich the user profile with valuable information without compromising on GDPR principles.

The insights, however, are so far only verified in the context of ecommerce. Other websites with different setups in terms of topology, or usage (with extra login and/or security) might benefit from using more specific information such as IP and location.

6. Conclusion and Future Work

This work elaborates a web page visit fingerprinting approach using sequence mining and compares it to other session stitching approaches tailored to web logs gathered from the domain of ecommerce. This new approach allows to utilise seemingly disjoint data, minimize challenges including but not limited to GDPR, and divide actionable insights to support decision-making on session matching which feeds into further marketing and web site analysis. It was empirically shown that with a relatively low number of sequential features, it is possible to obtain a good predictive result to identify matching sessions of the same users which is privacy-insensitive compared to the strongly revealing user-agent features that allow to uniquely identify sessions from the same

users. Furthermore, it outperforms embedding-based approaches while retaining a strong level of interpretability of the generated features. This was further illustrated by a number of managerial implications which were drawn from the result and aid decision-making for ecommerce stakeholders.

In future studies, we aim to look into larger datasets over long periods of time, potentially in combination with recurrent neural network-based approaches to capture the change in sequential behavior of visitors. We plan to apply the proposed approach to different contexts, e-commerce structures and industry sectors to demonstrate its genaralizability. Furthermore, we intend to investigate the influence of parameters further, more specifically whether stronger requirements on user visits helps further identify matching. Finally, we intend to investigate the use of dimensionality reduction techniques in combination with regular user-agent variables which can be combined with the sequential features.

Acknowledgements

We would like to thank the Edinburgh Parallel Computing Centre (EPCC) for their support to this research, as well as the DataLab for their funding.

References

[1] S. Kim, N. Kini, J. Pujara, E. Koh, L. Getoor, Probabilistic visitor stitching on cross-device web logs, in: WWW, ACM, 2017, pp. 1581–1589.

[2] Y. Qiao, Y. Wu, Y. He, L. Hao, W. Lin, J. Yang, Linking user online behavior across domains with internet traffic, J. UCS 24 (3) (2018) 277–301.

[3] X. Shen, B. Tan, C. Zhai, Privacy protection in personalized search, in: ACM SIGIR Forum, Vol. 41, ACM New York, NY, USA, 2007, pp. 4–17.

[4] R. Jones, R. Kumar, B. Pang, A. Tomkins, " i know what you did last summer" query logs and user privacy, in: Proceedings of the sixteenth

ACM conference on Conference on information and knowledge management, 2007, pp. 909–914.

[5] A. Gervais, R. Shokri, A. Singla, S. Capkun, V. Lenders, Quantifying web-search privacy, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 966–977.

[6] A. Cooper, A survey of query log privacy-enhancing techniques from a policy perspective, ACM Transactions on the Web (TWEB) 2 (4) (2008) 1–27.

[7] S. Goldberg, G. Johnson, S. Shriver, Regulating privacy online: The early impact of the gdpr on european web traffic & e-commerce outcomes, Available at SSRN 3421731 (2019).

[8] D. Jin, M. Heimann, R. A. Rossi, D. Koutra, node2bits: Compact time-and attribute-aware node representations for user stitching, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 483–506.

[9] T. Mielikainen, On inverse frequent set mining, in: Workshop on Privacy Preserving Data Mining, Vol. 1, Citeseer, 2003, pp. 18–23.

[10] J. De Smedt, G. Deeva, J. De Weerdt, Mining behavioral sequence constraints for classification, IEEE Transactions on Knowledge and Data Engineering (2019).

[11] H. Köpcke, E. Rahm, Frameworks for entity matching: A comparison, Data Knowl. Eng. 69 (2) (2010) 197–210.

[12] U. Draisbach, F. Naumann, A comparison and generalization of blocking and windowing algorithms for duplicate detection, in: Proceedings of the International Workshop on Quality in Databases (QDB), 2009, pp. 51–56.

[13] W. W. Cohen, P. Ravikumar, S. E. Fienberg, et al., A comparison of string distance metrics for name-matching tasks., in: IIWeb, Vol. 2003, 2003, pp. 73–78.

[14] A. K. Elmagarmid, P. G. Ipeirotis, V. S. Verykios, Duplicate record detection: A survey, IEEE Transactions on knowledge and data engineering 19 (1) (2006) 1–16.

[15] J. Wang, G. Li, J. X. Yu, J. Feng, Entity matching: How similar is similar, Proceedings of the VLDB Endowment 4 (10) (2011) 622–633.

[16] G. A. Wang, H. Atabakhsh, H. Chen, A hierarchical naïve bayes model for approximate identity matching, Decision Support Systems 51 (3) (2011) 413–423.

[17] M. Pershina, M. Yakout, K. Chakrabarti, Holistic entity matching across knowledge graphs, in: 2015 IEEE International Conference on Big Data (Big Data), IEEE, 2015, pp. 1585–1590.

[18] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[19] H. Zhu, R. Xie, Z. Liu, M. Sun, Iterative entity alignment via joint knowledge embeddings, in: IJCAI, ijcai.org, 2017, pp. 4258–4264.

[20] S. Punjabi, P. Bhatt, Robust factorization machines for user response prediction, in: WWW, ACM, 2018, pp. 669–678.

[21] C. J. Riederer, Y. Kim, A. Chaintreau, N. Korula, S. Lattanzi, Linking users across domains with location data: Theory and validation, in: WWW, ACM, 2016, pp. 707–719.

[22] M. Khan, N. Krishnamoorthy, L. Jalali, R. Biswas, Adobe identity graph, in: BigData, IEEE, 2018, pp. 5354–5356.

[23] T. Safavi, A. Fourney, R. Sim, M. Juraszek, S. Williams, N. Friend, D. Koutra, P. N. Bennett, Toward activity discovery in the personal web, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 492–500.

[24] S. Shekhar, D. Pai, S. Ravindran, Entity resolution in dynamic heterogeneous networks, in: Companion Proceedings of the Web Conference 2020, 2020, pp. 662–668.

[25] Y. Caspi, D. Simakov, M. Irani, Feature-based sequence-to-sequence matching, International Journal of Computer Vision 68 (1) (2006) 53–64.

[26] Y. Huang, J. Hsu, Mining web logs to improve hit ratios of prefetching and caching, Knowl.-Based Syst. 21 (1) (2008) 62–69.

[27] A. Guerbas, O. Addam, O. Zarour, M. Nagi, A. Elhajj, M. J. Ridley, R. Alhajj, Effective web log mining and online navigational pattern prediction, Knowl.-Based Syst. 49 (2013) 50–62.

[28] Y. Li, H. Wang, H. Gao, Efficient entity resolution based on sequence rules, in: International Conference on Computer Science and Information Engineering, Springer, 2011, pp. 381–388.

[29] S. Chaudhuri, V. Ganti, D. Xin, Mining document collections to facilitate accurate approximate entity matching, Proceedings of the VLDB Endowment 2 (1) (2009) 395–406.

[30] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, Deep learning for entity matching: A design space exploration, in: Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 19–34.

[31] Y. Qiao, Y. Wu, F. Duo, W. Lin, J. Yang, Siamese neural networks for user identity linkage through web browsing, IEEE transactions on neural networks and learning systems (2019).

[32] R. Agrawal, R. Srikant, Mining sequential patterns, in: ICDE, IEEE Computer Society, 1995, pp. 3–14.

[33] M. J. Zaki, SPADE: an efficient algorithm for mining frequent sequences, Machine Learning 42 (1/2) (2001) 31–60.

[34] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M. Hsu, Mining sequential patterns by pattern-growth: The prefixspan approach, IEEE Trans. Knowl. Data Eng. 16 (11) (2004) 1424–1440.

[35] J. M. Fowkes, C. A. Sutton, A subsequence interleaving model for sequential pattern mining, in: KDD, ACM, 2016, pp. 835–844.

[36] E. Egho, D. Gay, M. Boullé, N. Voisine, F. Clérot, A user parameter-free approach for mining robust sequential classification rules, Knowl. Inf. Syst. 52 (1) (2017) 53–81.

[37] D. Nguyen, W. Luo, T. D. Nguyen, S. Venkatesh, D. Q. Phung, Sqn2vec: Learning sequence representation via sequential patterns with a gap constraint, in: ECML/PKDD (2), Vol. 11052 of Lecture Notes in Computer Science, Springer, 2018, pp. 569–584.

[38] M. Pesic, H. Schonenberg, W. M. Van der Aalst, Declare: Full support for loosely-structured processes, in: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), IEEE, 2007, pp. 287–287.

[39] M. Westergaard, C. Stahl, H. A. Reijers, Unconstrainedminer: efficient discovery of generalized declarative process models, BPM Center Report, No. BPM-13-28 207 (2013).

[40] C. Di Ciccio, M. Mecella, A two-step fast algorithm for the automated discovery of declarative workflows, in: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2013, pp. 135–142.

[41] Y. Goldberg, O. Levy, word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method, arXiv preprint arXiv:1402.3722 (2014).

[42] W. C. McDowell, R. C. Wilson, C. O. Kile Jr, An examination of retail website design and conversion rate, Journal of Business Research 69 (11) (2016) 4837–4842.