UPPSALA
UNIVERSITET

IT 16 076
Examensarbete 30 hp
Februari 2017

Exploration and evaluation of different sessionization methods in a music streaming context

Martin Håstad

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Exploration and evaluation of different sessionization methods in a music streaming context

Martin Håstad

Sessionization is a powerful method to create aggregated data used to understand user behavior in the field of data mining. A lot of research has been performed over the years and most of it is geared towards traditional web servers. How the results translate to a context with a stand alone application with a service that has very specific usage pattern has not been fully investigated. In this project we investigated how the currently utilized heuristics perform on identifying a session in a music streaming context and introduce some additional measures to summarize such sessions.

Handledare: Boxun Zhang
Ämnesgranskare: Matteo Magnani
Examinator: Mats Daniels
IT 16 076
Tryckt av: Reprocentralen ITC

Contents

Chapter 1

Introduction

The amount of stored data in the world is ever rising. In fact it is estimated that there are around 1.3 zettabytes[1] stored globally [1]. It would take over 300 years to be able to only read all that data on a regular home computer. Fortunately this is divided over the entire world but large tech companies still have very large amounts of data sitting around on servers. The amount is so vast that analyzing it all has become close to impossible. However, where there is a need a solution appears and the answer to this particular problem is data mining. Data mining is a field which focuses on extracting patterns out of large amounts of data, utilizing a variety of ways, some which have been perfected and some which have just begun to be explored.

One of the more developed areas is association mining, which is used by e-commerce stores and large chain stores in order to identify which wares are often purchased together. In the context of e-commerce this gives the company a chance to give more tailored suggestions to users, in similarity to chain stores which can place items that are frequently bought together close to each other. Other notable uses are detection of credit card fraud or the exploration of DNA sequences. In all likelihood there is a lot more valuable information to be extracted from the data which has yet to be analyzed.

One of the techniques that has been developed to analyze web-traffic is sessionization, which is separating user's actions, during pre-processing, into a collection of items that are performed during one use. This is a subject that is well explored for web-pages where the available data is fairly homogeneous between different services. For stand-alone applications it has not been verified that the current heuristics are as well working, or which summarizing metrics are the most useful to summarize a session. This report

[1] Based on the estimated 295 exabytes in 2007 and the observed 23% growth a year

focuses on these questions in a music streaming context on the behalf of Spotify, aided by their data and knowledge.

1.1 Spotify

Spotify is a music streaming service started in Sweden in 2006 by Daniel Ek and Martin Lorentzon in collaboration with a few major record labels. It is available in 59 countries and has around 100 million active users [2]. This makes them the largest music streaming service considering the number of active paying users.

Spotify have begun focusing on data-driven development, which means that all business decisions should be based on data rather than intuition or personal experience. This requires data relevant to the business of high quality to be extracted.

1.2 Data Mining

As earlier mentioned data mining is the science of extracting interesting information out of large quantities of data. Usually the mining is divided into three steps which are essential to the process: data collection and preprocessing, pattern discovery, and pattern analysis [3].

Data collection is fairly self explanatory as it consists of collecting data that contains the interesting information. Then in the pre-processing phase the unwanted data or irrelevant data is filtered out.

The pattern discovery consists of performing statistical measurements or machine learning (ML) algorithms on the data in order to extract interesting information. The choice of method depends heavily on what patterns are of interest to the miner.

In the final step the patterns found are refined and used to generate a model for the desired information.

The rise of the world wide web has generated a whole new source of data where servers can collect information about the users using a service, which is the focus of web usage mining.

1.3 Web Usage Mining

Web usage mining is a sub-discipline of data mining which focuses on analyzing and understanding different aspects of user behavioral patterns in on-line

services. In practice the steps of traditional data mining applies, but due to the high similarity between many cases in the field the steps are specialized.

Data collection consists of collecting available data of how users interact with a service, which can consist of a range from which resource a user requested with a time-stamp to very detailed information of every click and small interaction with that service. In most cases pre-processing consist of removing malformed data or inactive data, for instance when a user requests a web page that in its turn can result in additional multimedia resources being requested which can be viewed as irrelevant as the user did not explicitly request them.

Sessionization is a method which groups the data into relevant chunks which can be performed during pre-processing, it will be discussed extensively in section 1.5 since it is the focus of this project. It is useful in order to have chunks of data with more meaning attached to them compared to individual data-points. The major problem that one is faced with is defining what constitutes a session and identifying such a group of data. Once it is known which data-points belong in the same session there is also the task of summarizing them in the most compact way while loosing as little information as possible. Performing data mining on top of sessions is commonly more efficient to understand high level user behavior rather than performing mining on all the raw data-points.

The common pattern discovery tasks includes analyzing which resources are the most popular and clustering users or sessions in order to look at common patterns.

The information that is desired in the end is similar to that of regular data mining, but some of the more specific areas are to predict some property or behavior. One such example is predicting if a session will result in a purchase.

1.4 Ethics

When performing data mining on real user data it is crucial that the data of individual users is protected. In order to ensure this all data is properly encrypted. Also user names are not included in any place where it is not necessary and instead another unique identifier is used. It has proved that in some cases this is not enough. An example is when Netflix made part of their anonymized data public a few researchers managed to identify users by utilizing external sources of information like the IMDB database of users movie ratings. In other words keeping the number of such datasets in circulation to a minimum is essential to ensure user integrity. All of this has

been kept as the highest priority during the project and report writing, and is enforced by the central organization by enforcing encryption and having guidelines on how to handle data.

1.5 Problem structure

Currently the heuristic *h2* is being utilized at Spotify to separate data into sessions. For more details on this algorithm see section 2.1. It has never been been properly evaluated how well working this approach is or if some other method would generate more accurately constructed sessions. The second property that needs to be investigated is which metrics[2] that accurately summarizes a session. The objective of the thesis is to investigate these properties.

Most research in the web usage mining focuses on performing analysis on website services. In the context of a stand-alone application a different set of metrics are available, which provide more information both to be used as summarizing metrics and to determine what constitutes a session.

Grouping data solely based on which time a certain event occurred could be a too crude approach in these cases. By utilizing more information, an investigation was made if it were possible to create more well defined groups of data that generate better data mining results. The challenge is that the amount of data is vast and in order to be a viable approach it must be possible to define a session in a computationally efficient way. It is also crucial not to introduce bias into the data during the pre-processing, that could affect the end mining. For example if the session construction takes the artists played into account this could introduce bias. This due to the fact that if any conclusions are to be drawn in later stages about which artists are played in the same session the result would be affected by this pre-processing decision.

Traditionally sessions are defined as all actions that one user performs during one use of a service. A problem in the context of a stand-alone application is that it is common to leave the application running, either playing music passively or the user present. Distinguishing between these two cases is desirable, but not simple. Also it is possible that user behavior changes during a single use, for instance if a user is passively listening to music and decides to start browsing for new music. It might be possible to distinguish between those two behaviors with enough accuracy to handle

[2]By a metric we here mean any type of information that can be used to identify a session or data-point.

them in a different way to improve the descriptive properties of a session. Even more significant is that this might identify if the physical user changes as people could be using the service on a shared computer.

1.5.1 Evaluation

One of the problems is that understanding what constitutes a well formed session is difficult without the actual data from users where each session begins and ends. If this data was known it would be possible to compare how well an attempted solution agrees with the correct solution and through this compare different solution attempts.

Keeping in mind that performing data mining on the sessions is the end goal we can use this property to evaluate a session. There are several models available at Spotify that utilizes sessions in order to perform predictions. It is then possible to train these model on the different partitions of sessions created by the different solution methods and compare which one generates the most accurate predictions. In all likelihood the result will be more stable if several models are used in the comparison as opposed to a single one.

This approach evaluates both how well formed the sessions are as well as how relevant the summarizing metrics are. Unfortunately it is difficult to separate the two. It is, however, possible to vary only one of the properties for every trial in order to get some measurement of the different qualities.

It is also possible to attempt to manually verify how well constructed sessions are. This can be utilized to spot obvious mistakes, but the data mining approach is more refined as it shows that the method is well suited for the end goal.

Chapter 2

Background

2.1 Current approaches to sessionization

There are three different heuristics commonly used to separate data into sessions. They are in most literature referred to as *h1*, *h2* and *h-ref* [3][4]. The *h1* heuristic defines a session in such a way that all actions of a user that happen within a specified threshold of time after the first action of a session belong to that session. The second heuristic *h2* is also time based, but rather defines that two actions belong in the same session if the time difference between them is below a certain threshold.

The last one *h-ref* is more complex in many cases since it defines that two actions a_i and a_{i+1} belong to the same session if performing a_i leaves the user at state where it is possible to perform a_{i+1}. For web pages this means that if you load two pages in a row, those page loads are considered to belong to the same session if there is a hyper-link from page one to page two. This is, however, made more difficult by the fact that an entry might be missing somewhere in the data which can for instance occur due to a bug or that the client had some parts of the path cached. It is common that *h-ref* also has some timeout threshold.

One possible improvement on the *h2* heuristic is to have different values for the threshold after different actions [5]. This is beneficial due to that a user stays in contexts a different period of time depending on the content of that context.

People have also experimented with non-greedy algorithms such as applying integer programming to maximize a target function, given a function describing how different session constructions should be rewarded [6][7]. The runtime of such a method is of too high computational complexity to be of

interest in this case where it is usually possible to add more data instead to get better results.

2.2 Transaction identification

A transaction is a subset of records from a session which are more strongly linked than the session itself, usually defined as all actions leading to the same goal [8]. This makes it possible to split a session into several smaller transactions which by themselves form descriptive groups of data.

As in the case of session construction, the methods used in traditional web usage mining are not very relevant, as they rely heavily on page navigation patterns [9]. The concept is interesting to the project, since dividing sessions into more refined collections could be a promising method to increase the accuracy of future mining.

2.3 Modeling user burstiness

Many human actions are performed in a bursty fashion, which means that many actions are performed in quick succession after which a longer pause follows, rather than a constant size interval between actions [10]. Knowing that this property probably exists in sessions as well we can try to measure it in order to characterize user behavior in sessions more accurately than simply keeping track of the number of actions.

In order to properly model how such a signal behaves more information than the mean and standard deviation of the inter-arrival times is necessary. Since such information is only sufficient to get an idea of the average inter-arrival time as well as how much it varies, not in what order or fashion the actions arrive. A measure has previously been proposed to model these patterns [11]. It consists of two variables which are calculated by looking at n inter-arrival times τ_i, $1 \leq i \leq n$. The first factor is the burstiness $B \in (-1, 1)$ where a value of -1 indicates a highly regular signal, 0 a neutral signal and 1 a bursty signal. in practice the value -1 is achieved when the standard deviation of inter-arrival times is 0, 0 is achieved when the standard deviation is equal to the mean, and 1 when the standard deviation approach infinity. It also utilizes a memory variable $M \in (-1, 1)$ where a value of -1 indicates that a long inter-arrival time is usually followed by a short and vice versa, and a value of 1 indicates that a long inter-arrival time is usually follow by a long one and the same for short inter-arrival times. The variable

M represents the autocorrelation of the inter-arrival times with a distance one between them, which is a common metric to discover patterns in signals.

They are calculated in the following fashions:

$$B = \frac{\sigma_\tau - m_\tau}{\sigma_\tau + m_\tau}$$

$$M = \frac{1}{n-1} \sum_{i=1}^{n-1} \frac{(\tau_i - m_1)(\tau_{i+1} - m_n)}{\sigma_1 \sigma_n}$$

Where m_τ is the mean of all inter-arrival times and σ_τ the experimental standard deviation. The variables m_1 and σ_1 represent the mean and standard deviation of $\tau_1, \tau_2, \ldots \tau_{n-1}$, and similarly m_n and σ_n represent the mean and standard deviation of $\tau_2, \tau_3, \ldots \tau_n$.

2.4 Models

Before diving into the project specifics, knowledge of a few complex models are needed. These are necessary in order to work with some features and evaluate the project.

2.4.1 Markov Models

In order to be able to look at different state transitions of more complex fields a model to represent the different behaviors is needed. A Markov model is built on the assumption that the only property that influences the probability of being in state i at step s is the state at step $s-1$. Thus the model contains the probability to transition to all possible states for each possible state. This is usually represented as a matrix where an element $e_{i,j}$ represents the probability to transition from state i to j.

2.4.2 Hidden Markov Models

A Hidden Markov Model (HMM) is instead based on the assumption that what is observable at step s is dependent solely on a hidden state. So the model consists of n hidden states with probabilities to transition from hidden state to hidden state. Also each state has a certain probability of generating a certain output. Given a sequence of outputs there are many available methods to train such a model, which are already implemented. How these work is out of the scope of this report. Such a model generates hidden states, but what that state represents, for instance that the sun is shining or that

it is raining, is not always deducible. With an already generated model it is possible to predict which underlying hidden state generated the output, as well as the overall probability of such a sequence being generated.

2.4.3 Decision tree classifiers

A decision tree classifier is a model used in data mining to classify a record of a dataset based on the metrics of that record. The reason it is called a decision tree is that it is built as a tree and in every node a value of the record is used to determine if it belongs to the left or right subtree of that node. The conditions that exists in these nodes are of the types less than, greater than or equal to a constant value. Then the same process is conducted for the root node of that subtree. This process in repeated until a leaf node is reached which contains the guessed class. There are many different algorithms to train this model from an existing set of data. These are out of the scope of this project and will not be discussed further.

In this project only one metric will be used to classify. With only one metric the training process is quite simple since a single split is probably the optimal, of course the metric could be split into different ranges which generate different predictions but that would in all likelihood be a symptom of over fitting.

Chapter 3

Method

3.1 Data exploration

In order to be able to work with the raw data it is necessary to know what metrics are available and the quality of the data. Thus step one consisted of working with the data, identifying potential problems, and looking at session trends manually. Potential problems turned out to be many, including missing or malformed data and that different versions of the application report different sets of data.

It was decided to focus mainly on data from a limited period of time and on a specific platform with few known issues where the data is fairly consistent in order to get a more reliable analysis.

What can be concluded is that there are quite a few interesting metrics that can easily be extracted from the data. For playbacks these include:

- Time-stamp - When the playback finished.
- User data - Data to identify which user started the playback.
- Playback duration - The total length of the playback.
- Percentage played - How much of the started playback a user played.
- Skipped playback - If the playback was interrupted.
- Context - Which collection or list that was used to start the playback.
- Seeks - The number of times the user jumped to another part of a song. (Joined in from a separate table)

- Artist - Which artist was played in the playback. (Joined in from a separate table)
- Shuffle On/Off - If shuffle is active in the context or not.
- Off-line - If the playback is part of an off-line session.

And for page-views:

- Time-stamp - When the song started playing.
- User data - Data to identify which user started the playback.
- Page content - Which type of page was requested.
- Time in view - How long time was spent on the page.

These metrics can also be combined in different fashions in order to get measurements that either describes sessions better or form a better model of user behavior. As an example we can look at the activity level of a user as a function of how many pages they visit, how many times they seek in a song and how often they skip a song.

In order to better understand the different types of sessions the heuristic *h2* was used to group the data. Since page views were not available for all platforms in this dataset they were ignored in order to get more accurate statistics from the playbacks. This is useful both to identify potential problems caused by the assumptions *h2* is built on, and to avoid having to look at the time-stamps which is time consuming, due to it being a complex field. Trying to manually identify places where the heuristic could be improved turned out to be problematic. Most sessions were fairly standard in the way that there was playbacks back to back and parsing such data is very time consuming and better left to an automated process. What could be identified, however, is that a few actions stem from some design choices in the application. The most common one being that the application starts in the same state as it was stopped and it is then common for the user to just start the last action to then immediately change song or even context. Trimming such entries might improve the quality of session data, which could be valuable to investigate in the future.

3.1.1 Investigating the raw metrics

After having the data separated into sessions, it is possible to calculate statistics of the different metrics. There are two interesting aspects to consider for

each statistic, namely how it varies within one session and how it varies from session to session. If a metric has a low variance in the intra session perspective compared to the variance in the inter session context it is a promising asset to separate different types of sessions.

We can identify that we have three different kinds of raw metrics, which require different methods to extract interesting information.

For values that are numerical how to calculate the mean, median, and variance is trivial since those measurements are well defined. Such variables are how often a user skips to another location in a playback, how much of a song is played on average or the average time between two playbacks.

Variables that are non-numeric require another approach to quantify the distance between two values, which is necessary to have a measurement of how they differ. These include which artists are played, which contexts are used and which pages are viewed. How to do this in the most accurate fashion is difficult to say and will be discussed later.

The off-line, shuffle and skipped metrics are binary but it is still interesting how they change over the course of a session. So it might be interesting to have a measurement that takes that aspect into account.

Having these metrics is valuable in two different ways, both to use as describing metrics for a session and to explore the change of the metrics during a session in order to attempt to identify a change in behavior.

Then when having defined the different metrics it was explored how they change when the threshold value of *h2* changes. This serves to both see how the metrics vary over the different sessionization attempts as well as giving us an idea of which threshold value for *h2* seems to be optimal.

3.1.2 Handling the binary metrics

It would be possible to simply assign numeric values to the different states of a binary metric in order to be able to calculate a mean and a variance. This model, however, would not differentiate between the cases where the variable is toggled for every other data-point and where it toggles once in the middle, despite these being very different behaviors.

Perhaps a more viable first approach would be to count the number of toggles for the shuffle and off-line characteristics, as well as the number of positive values. This would generate a metric which gives a lot more information.

If a user skips a song that is a very different and in general more common action than turning on or off shuffle. Here it might be more useful to look at the skipping behavior and try to model that, or tying it in with other

variables to model the activity level during a session. Which approach is better is not clear and both must be investigated. For the time being we will have the skip ratio as a simple measurement of user activity.

3.1.3 Quantifying the non-numeric metrics

The simplest way to get a number of these metrics is to count the total number of each in a session. Which for the current metrics mean how many contexts were active and how many pages were viewed.

A more refined method than just looking at the total number of distinct entities is desirable for the complex metrics in order to extract more information from them.

The different metrics require different methods to be compared since they represent vastly different properties. The distance between two artists can not be quantified in the same way as the distance between two page views.

Contexts and viewed pages

In order to be able to get a measurement of the distance between two contexts or pages viewed two methods were used and compared, namely a Markov model and a HMM of the transitions of contexts.

Markov models operate on the assumption that the only property affecting the probability of different values for state s_{i+1} is the value of s_i. This assumption is probably more true for viewed pages since what page is currently viewed limits which pages are accessible, and also there might be some common paths. Which play-context that is active is heavily dependent on which pages have been viewed for instance.

HMMs on the other hand assumes that the probability for a value in state s_i depends on a hidden state which is not directly observable. What the hidden states represent is in many cases difficult to say. And how to utilize them will depend on what models are constructed. They might describe the behavior better since the assumption is more sound that the behavior depends on something that is not directly observable, for instance user mood or current company or something similar.

With the Markov model it is possible to calculate how typical a context switching behavior is by simply multiplying the different transition probabilities with each other. The unfortunate result of this is that a longer browsing session will in general have a lower typicality measure. This effect can be counter-acted by dividing the measure by the value of the most likely possible switching behavior.

3.2 Investigating the current heuristic

Currently Spotify utilizes the heuristic *h2* with a threshold value of 15 minutes. The first natural question is if any other heuristic could potentially outperform *h2*. Both *h1* and *h-ref* are poorly suited to deal with the situations that arise in a music application. The heuristic *h1* requires a time-out which represents the maximum length of a session, which in this context could be very long and it would thus risk misclassifying the shorter sessions. Also *h-ref* is unsuitable due to there being a limited amount of browsing in the application, and there is no way to skip to a page which is not directly reachable from the current page.

After excluding the possibility to use another common heuristic the logical step is to see if any simple improvements can be made to the existing one. The most obvious one is to verify that a 15 minute threshold makes sense, or if other values produce better results. The other is the possibility of having a variable threshold as discussed when introducing the heuristic. It would make sense if the time between the start of two playbacks varies with the length of the playback in between. This method potentially have the advantage of better capturing sessions with very long playbacks as well. Let us call the modified heuristic *h2'*. Which also is dependent on a threshold value but rather than comparing a playbacks inter arrival time with the threshold value it is compared to the threshold value + the length of the playback.

In order to investigate which values or method seems to work best we will look at summary statistics over all sessions created by a heuristic in a subset of the available data, and see if any trends appear. Then investigate a few points of interest further with the method described in section 1.5.1. The reason for trying to evaluate a few points rather than all is that performing a full scale test is quite resource demanding.

In order to be able to work with larger sets of data by avoiding joining with other tables it was decided to drop the artist and seeks fields at this point.

3.3 Analyzing alternative approaches

A manual verification of the current approach was also conducted in order to see if any obvious mistakes were by the different heuristics, and how these could be identified. However, determining any obvious faults in the current heuristics turned out to be more problematic than anticipated, overall the

sessions were well formed. Thus the search for a heuristic utilizing more data than the time-stamp was abandoned and it was decided to instead try to identify transactions within the sessions. Transactions have the potential to describe user behavior better than full sessions.

In order to identify transactions the concept must be defined for the specific context of streaming music. Traditionally the definition is that all actions that strive to the same goal belong to the same transaction which is harder to define in this context. It is perhaps viable to define a spread for a session, a measurement of how scattered the behavior is within a session. The same measurement can be used for transactions.

Then it is a matter of splitting sessions into transactions such that the transactions have minimal spread. Good splitting points would thus be at a point in a session where user behavior radically changes. As stated a transaction is defined as all actions striving to the same goal, in a music streaming context this could for instance be when a user transitions from actively trying to discover new music to passively listening to a play list. Which would be caught by splitting in the way described. A session or transaction of size one will according to most definitions have an spread of zero. However, creating too small transactions is undesirable as it would approach the same setting as working with raw data. So some method that punishes small transactions is necessary.

As a proof of concept we will start by looking at sessions that with strong confidence can be split into two transactions. Choosing if and where to split a session is a significantly simpler task than investigating and choosing multiple such points.

The idea is to split some of the sessions into transactions where applicable, and defining the others as single transactions. Then it is possible to performing mining based on these new chunks of data and analyze if it yields better mining results. If that is the case it shows that identifying transaction is a promising approach to improve the usefulness of sessions.

3.3.1 Defining session spread

In order to define a spread we need measurements of the spread for the available and relevant metrics. Again for metrics with a numerical value it is fairly straightforward as it is easy to calculate the variance or standard deviation to get an idea of how much it varies throughout the session. Other metrics would need to be handled in a different fashion.

When just looking for a single optimal split we can utilize a simpler definition of spread to accomplish this. For each possible time-stamp within

a session we split around that time-stamp and look the distance between the summarizing metrics for each transaction, and define the largest distance as the spread. Since this is only a proof of concept and not an attempt to find the optimal value looking at the naive statistics is sufficient with a few modifications. For the pattern of viewed pages for instance the generated HMM , see section 5.4 for further details, was utilized to analyze how the hidden state changes. The measurement of distance between two sets of states, lets call them A and B, was defined as:

$$1 - \frac{|A \cap B|}{min(|A|, |B|)}$$

The reason for defining the distance in this fashion and not with something more traditional like the Jaccard index, which is the size of the intersection divided by the size of the union of two sets, is the fact that it is to be utilized to split into two transactions with distinctly different patterns. Thus the quality that the distance is 0 if $A \subset B \vee B \subset A$ is desirable, while maintaining distance 1 if $A \cap B = \emptyset \wedge A \neq \emptyset \wedge B \neq \emptyset$. The same was done for the play contexts utilizing the matching HMM described in section 5.5.

It would also have been possible to use the Markov Models to look at how common a transition is in order to define distance in that fashion, but it is problematic that a transition into any rare state will be considered to have high spread in such a case. This effect could be counteracted but using the HMM:s seemed more appropriate to identify changes in hidden user states, since that might indicate an overall change in users mindset.

Furthermore the time spent in a view was normalized by dividing with the average time spent in all views of that type with a fixed maximum timeout of two minutes to be able to compare the different views.

3.3.2 Evaluation

To evaluate the different methods it was decided to use a simple ML model in order to minimize eventual noise. After some discussion it was decided to predict if a user will use the service in the last two weeks of a month given the data of the first week in the same month. There is existing knowledge that the total number of sessions is a good property to use to predict such qualities. In the end evaluation this was used as the sole feature for the ML model, after trying a few models with more features. Then train this model on a set of data and see how well on average it correctly classifies a lot of different evaluation sets.

The reason for using such such a simple approach is that if a more advanced model with a lot of features are added the end result might be overshadowed by the performance of the model. Furthermore using this model makes it possible to try more different models and to perform repeated tests to a greater extent in order to get more accurate results due to the low computational complexity.

3.4 Investigating summarizing metrics

Creating a stable method of evaluating summarizing metrics turned out to be more difficult than anticipated. When using the above described model adding even basic metrics such as average session length did not influence the result significantly. Instead a manual evaluation of how well the information from the raw data is retained in the summarizing metrics and what information it does not capture will be conducted.

3.5 Explaining the early spike

While testing different threshold values for the *h2* heuristic an unexpected spike was found for very low threshold values. A guess could be that it differentiates between active and passive sessions as it treats them differently. As discussed earlier it is very hard to determine if a stream that start automatically after another stream is finished is actually part of a session or not, and so far it has been assumed that a new stream forms a new action as the user lets it continue playing which could be regarded as a passive action. Due to this spike it is worthwhile to investigate how the heuristics behave if a stream that is played all the way through with no action is treated as not an action instead.

In order to do this the heuristics will have to be modified slightly but otherwise the same procedure can be utilized. With this approach the streams that are not close enough to a previous action will have to be regarded as data generated automatically by the application and disregarded.

Chapter 4

Implementation

In all of data mining it is crucial with efficient algorithms, since the amount of data generally is large. Due to the scope of the project not all code was optimized to the full extent possible. When it is for investigation purposes it is more important with code that produces a correct result, rather than generating correct results in the most efficient way possible. Already created tools were also used to the largest extent possible in order to minimize programming errors and save time.

4.1 Tools

The following tools were primarily utilized to complete this project:

Python An interpreted programming language, with many built in features and available packages.

Cython A compiler which allows C extensions to be written in and used from Python.

NumPy A package containing primarily a powerful N-dimensional array object. Also supports vector operations.

Pandas Pandas is a Python framework that works efficiently with large sets of data. Utilizes NumPy for storage.

Jupyter Notebook Environment for writing Python code with a simple graphical layout.

Sci-kit A Python package containing a wide variety of ML algorithms.

To be able to work with the amount of data that it is interesting in the data mining context Pandas is almost a must. In general one gigabyte of data-points were loaded in at a time. Pandas also contains a lot of features to work with the data it has loaded, but for more in depth analysis where access to several parts of a record at once was necessary turned out to be to slow. The solution was to access the underlying NumPy arrays directly in Cython to be able to look at the records efficiently.

4.2 General process

The general work flow was to load the data into a pandas data frame, which is the pandas default storage, in a Python environment. Then to pass the underlying NumPy arrays to a Cython function which performs the desired calculations and returns the result back to the Python environment. Then the result was interpreted in the Python environment with the benefit of being able to easily explore the data and create the desired plots. All Python and Cython code was written in the Jupyter Notebook environment, where it was possible to mix the two without effort.

4.3 Implementing different sessionization methods

In order to investigate the different heuristics a relatively simple approach was used. First the data was sorted after the unique identifier for each user and then by time. This could be accomplished efficiently with Pandas in the Python environment. The sorted data was then passed to a Cython function. Then for each threshold of interest the sessions were formed by looping through the data looking at if the time difference was larger than the threshold or if the user changed. The relevant data, as discussed previously, were temporarily put in lists for each session. From these lists the end metrics of the session was calculated and stored. The calculations for sessions where the length of the playback were factored in were made in the same fashion, but with an updated calculation if the time difference exceeded the threshold.

4.4 Investigating complex metrics

The HMM:s were trained and utilized by using a Python library called hmmlearn. The library contains ready made methods to train HMM:s from several sequences of outputs. The training consisted of giving each possible value a unique integer value and separating the raw data into sessions using the

current heuristic *h2* with a 15 minute threshold. This heuristic was used due to it being the best known solution at the time, and it is necessary to group the data-points in order to understand what happens within one session. If all data-points were fed in one stream the resulting HMM:s would not represent behavior in sessions. Then several HMM:s were trained on part of the data and evaluated on another set and the model that had the highest probability of generating the evaluation set was saved. This was done for a different number of hidden states. Then a manual analysis was conducted to identify the most relevant HMM for the project. This was done for both the feature used and views.

The Markov Models were implemented in Cython, with a two dimensional NumPy array to represent the transition matrix. Where element $e_{i,j}$ represents the probability of transitioning from state i to state j. To fill this matrix all sessions were fed into the model in the fashion that if a transition $i \rightarrow j$ is observed within a session then $e_{i,j}$ is incremented one. Then all elements in a row is divided by the total sum of the row in order to create probabilities.

4.5 Finding the optimal split

In order to find the split in a session with the highest spread as described previously it was necessary to be able to find it by just looping through each created session once. Anything above linear complexity will severely reduce the amount of data that can be analyzed, which is undesirable in the pre-processing step.

In order to achieve linear complexity a slightly better approach than the naive was necessary. Once a session was created, with the summarizing metrics calculated, the session was looped through once again and each data-point was in order of time-stamp added to a separate transaction and removed from the current. After one data-point was moved the distance between these two transactions was calculated. Since we are only interested in the highest distance achieved only that value was saved.

Also it was discovered that a lot of the highest distances were achieved when a single data-point at the beginning or end of a session with extreme values formed a transaction. This could be counteracted by either rewarding long transactions or declaring a minimum size of a transaction and for simplicity the latter was used.

Which distance to use was not quite clear so the euclidean distance was used with weights for all the different metrics. In order to understand which

weights were the optimal a manual evaluation of the sessions with the highest maximum spread was conducted. The weights were then slightly adjusted and the process repeated. When the sessions that looked promising to split were the ones with the highest distances it was necessary to decide which threshold for splitting to use. A fairly high threshold was chosen due to the fact that this is an attempt at proof of concept rather than finding the optimal solution. The metrics that were used to calculate the splits were:

- Average percentage played
- Skip ratio
- Average time in views (normalized as discussed above)
- Burstiness measure on all actions performed
- Shuffle ratio (Playbacks that ended with shuffle active)
- The HMM state distance (discussed above)

When the best splitting point was found and the distance was greater than the threshold the session was split into two transactions. The way this was implemented in the project was to give the two transactions different session IDs, while the others represent a single transaction and thus were given a single ID, in order to be able to work with the data in the same way as before.

4.6 Machine learning models for evaluation

In order to perform evaluation 100 datasets were taken and separated into sessions using a single heuristic. Then a subset of all users was taken from each dataset with users who had at least one stream during the first week, which on average were around 3500 users per dataset. These users were then classified into two different categories based on if they had streams in the last two weeks or not. Then a decision tree model was trained on 20% of these users chosen at random and the remaining 80% were used to evaluate the model by letting the model classify each user and then checking if the classification was correct or not. This was done for every set of data with a 100 runs on each dataset, resulting in 10000 runs. Then the average correct classification rate of all runs was calculated in order to score the sessionization method. The only metric associated with each user was the user's total number of sessions in the first week.

Chapter 5

Results

5.1 Data Exploration

The first step of investigation was to see what happens with different metrics of sessions depending on the threshold value of *h2*. The result can be seen in figures 5.1, 5.2, and 5.3.

There is a clear trend that session sizes grow as the threshold size increases which was expected. Also the standard deviation of the monitored metrics grow at about the same rate as the session size. The three different areas of interest can be said to be 5-10minutes where the session size is rapidly growing, 10-15 minutes where the growth rate is stabilizing, as well as the upper limit of 20-25 minutes. No other areas of clear interest appeared. Since the spans are quite broad an investigation of all points might as well be conducted.

Also a large change in the standard deviation of number of seeks between sessions can be seen. This was found to be caused by a single data-point with a few hundred backward and forward seeks, which could be attributed to either a logging error or an extreme user behavior.

Investigating the binary metrics further turned out to be of relatively low value since they were normally toggled zero or one times.

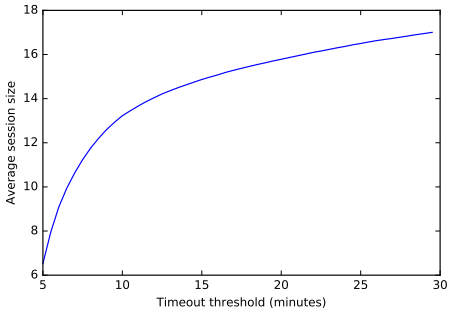

Figure 5.1: Average session size using different threshold values for *h2*.

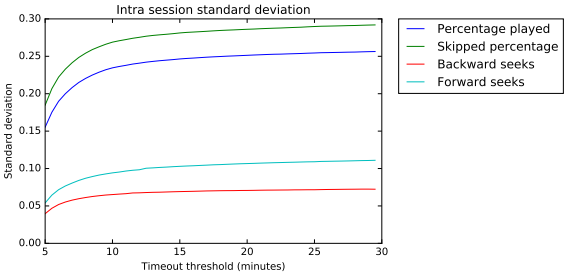

Figure 5.2: Average standard deviation of metrics in a session using different threshold values for *h2*.

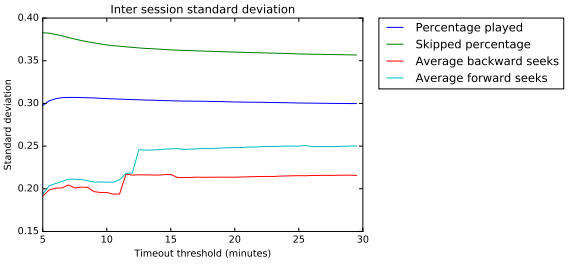

Figure 5.3: Standard deviation of average metrics between session using different threshold values for *h2*.

5.1.1 Alternative heuristics

The next step in the analysis was to investigate how the *h2'* heuristic described in section 3.2 performs. The difference being that it takes the length of the playback between two points into account.

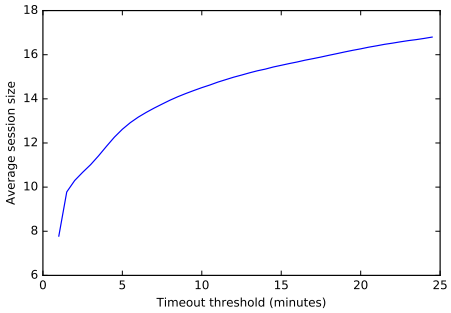

Figure 5.4: Average session size using different threshold values for *h2'*.

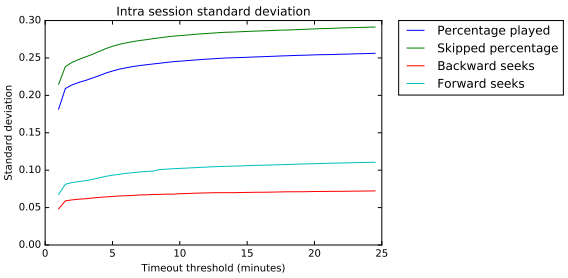

Figure 5.5: Average standard deviation of metrics in a session using different threshold values for *h2'*.

As can be seen the values seems to correspond closely to those generated by *h2* except that they have been shifted by about three minutes. This probably due to that being about the average length of a play-back. There is also a change in the curve for very low threshold values.

5.2 Transaction identification

In order to identify transactions different weights were needed for the different metrics, this was done by tuning the weights and manually evaluating

the transactions created. After many attempts the weights that produced the best result was:

Metric	Weight
Skip-ratio	0.3
Feature HMM	0.25
View HMM	0.2
Percent played	0.125
Burstiness	0.125

Shuffle ratio and the average time in view did not appear to give any extra information. The distance threshold was mostly set to 0.7 due to almost all sessions having a maximum transaction distance of at least 0.7 appeared to have a clear change in browsing behavior. Higher values were also tried when the transaction identification gave unsatisfactory results.

5.3 Evaluation

In order to evaluate the method described in section 4.6 was used to evaluate different thresholds for both *h2* and *h2'*. The threshold value was increased by 30 seconds at a time for both heuristics in order to find different peaks.

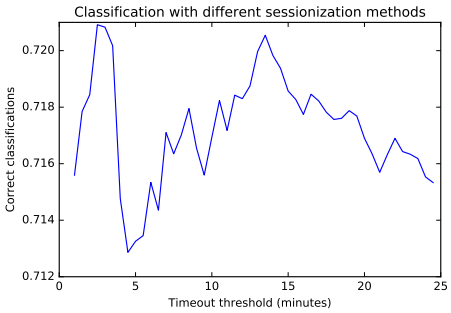

Figure 5.6: Average correct classifications using different threshold values for *h2*

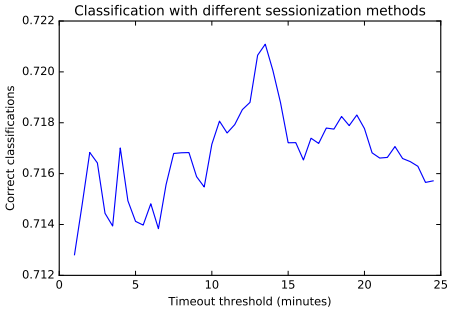

Figure 5.7: Average correct classifications using different threshold values for *h2'*

As can be seen the resulting graphs are similar, both having a peak at around a threshold of 14 minutes. This is kind of surprising as using different heuristics to calculate the splitting point was expected to skew the result in the same way as the session sizes in the previous figures 5.1 and 5.4. Those figures were calculated using only streams and did not include views which can explain the result. If a stream is started again at some point, then a page view is performed first which makes the length of the stream irrelevant. There is such a low amount of streams with a length that exceeds the threshold length that it does not appear to influence the end result in any significant fashion.

There is also a peak at 2.5 minutes for the regular heuristic which is surprising as most songs exceed that length and only active sessions where

the user regularly skips a part of each stream would form sessions of a length greater than 1. The passive streams would most form a separate session each.

An investigation of the different peaks were conducted as well with smaller changes in threshold values in order to get a better understanding of how the evaluation behaves at those points.

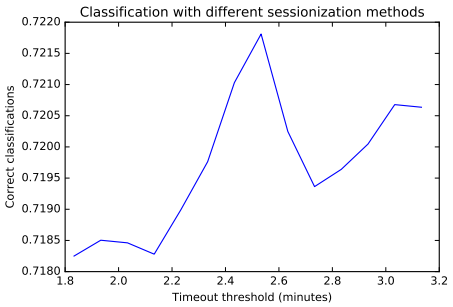

Figure 5.8: Average correct classifications using different thresholds for *h2*, zoomed around the first peak

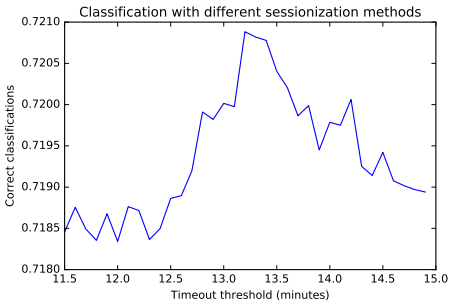

Figure 5.9: Average correct classifications using different thresholds for *h2*, zoomed around the second peak

First of all we can conclude that the trends seem quite clear which makes it more probable that the result is correct and it is not random noise, which could have been a problem due to the relatively low change in correct classification rate. The peaks that coincide appear to be at 13.2 minutes for both heuristics while the early peak using the *h2* heuristic appears at around 2.55 minutes.

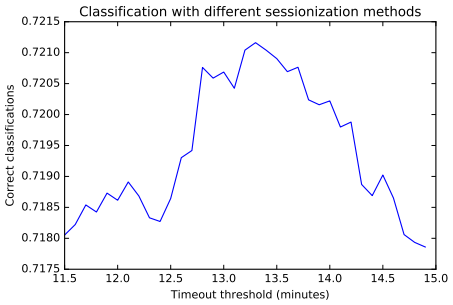

Figure 5.10: Average correct classifications using different thresholds for *h2'*, zoomed around the peak

Transaction identification was attempted on sessions created by the top performing heuristics. After transaction identification the mining result became significantly worse even for very conservative thresholds of the distance required to split. Only 71.18% of users were correctly classified which is lower than all other methods. After trying with different thresholds and splits the only clear trend was that the result got worse the more sessions that were split into transactions.

5.3.1 Disregarding passive streams

When repeating the evaluation step with the modification of regarding passive streams as non actions instead of actions the results turned out quite differently.

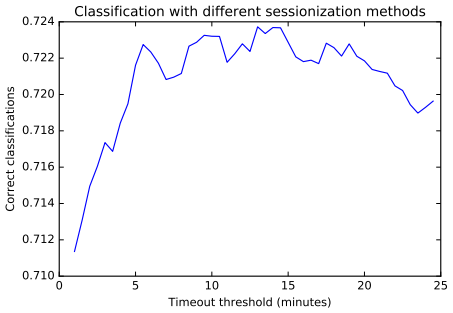

Figure 5.11: Average correct classifications using different thresholds for *h2*

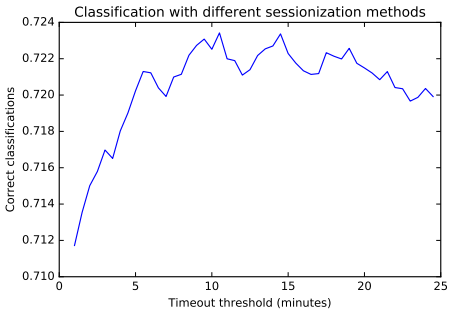

Figure 5.12: Average correct classifications using different thresholds for *h2'*

The clear pattern is that too short or long thresholds are bad while the results in the middle seem quite even with a small fluctuation, which is true for both graphs. Also the early spike appears to have been eliminated by the different definition of a session. As can be seen the correct classification rate was generally higher, although it is worthy to note that the difference is only around one per-mil points.

Transaction identification was also attempted on some of the top values here, where very few good splits were found and this actually improved the result very slightly, approximately 0.1 per-mil points more accurate predictions. With the current heuristic to identify transactions only 50 out of around 35000 sessions were split, which means only one out of 700 sessions are separated into transactions.

5.4 HMM for pages viewed

When training a HMM on the pages viewed the most interesting result appeared when having four hidden states. The four states were fairly "stable" meaning that it was unlikely that a user would transition from one to another, as can be seen in figure 5.13. Which was the most usable since it shows which pages are usually accessed together during a single session.

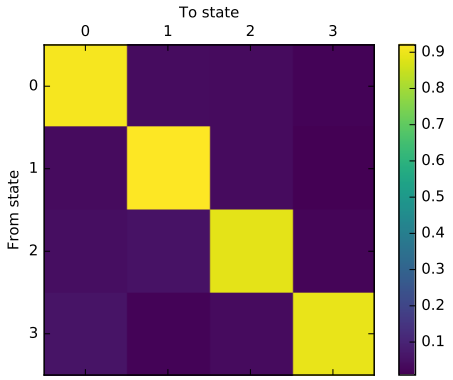

Figure 5.13: Transition probabilities for page view HMM

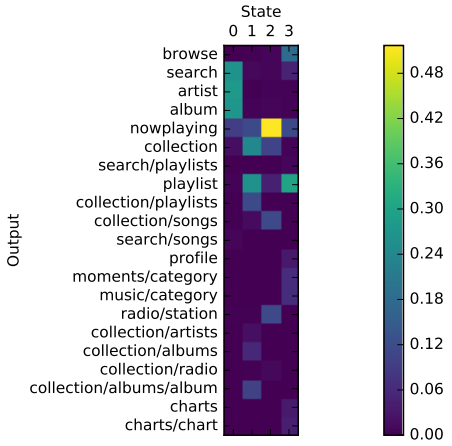

Figure 5.14: Output probabilities for page view HMM

We can look at the most common viewed pages in each hidden state, which can be found in figure 5.14, to try to get some idea of what they represent:

State zero: Search, album, artist and now playing

State one: Play list, different collections and now playing

State two: Now playing, collection, radio, collection/songs and play list

State three: A large variety of pages

5.5 HMM for features used

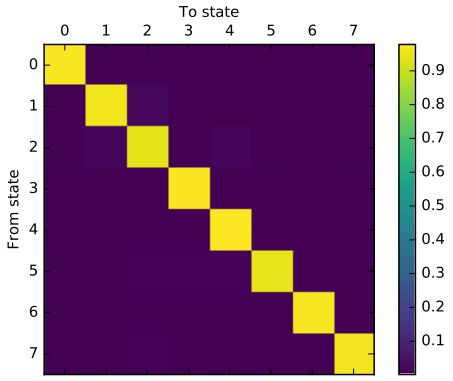

Figure 5.15: Transition probabilities for feature used HMM

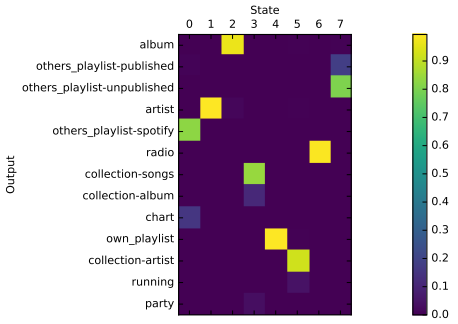

Figure 5.16: Output probabilities for feature used HMM

The HMM for the features used to play songs shows some additional resolution over where music is played from. We again chose a model where it is rare to transition from a hidden state to another hidden state. The data that can be easily extracted is that if a user starts playing music from a self made play list, a single artist, a single album or the radio they are very likely to continue doing so.

Chapter 6

Discussion

The overall conclusion is that the previous method is quite effective. From the data presented in the result it seems that the 15 minute window might be slightly long, and could be adjusted a bit in order to form more well formed sessions. However, since the project only investigated one platform the results might not translate to other platforms and they would need to be investigated separately. The most interesting result was going away from the passive streams as passive actions definition, as this yielded better predictions. This is an indicator that passive streams that are surrounded by long periods of inactivity should be regarded as data-points automatically generated by the application.

The practical effect this change would have had in the above prediction scenario would have been to separate sessions which have an initial active period, which all sessions do, followed by a longer period of passive streams and then another active period. Sessions with these kinds of patterns would usually not be caught by the attempted transaction identification, since they contain three different sections of behaviors. A proper transaction identification would perhaps separate such a session into three different transactions or more.

The transaction identification was a failure for all the attempts where the passive streams were included as active points. Probably the reason why is that a common session to split was very active ones that then were abandoned. Which seems intuitive to be counter beneficial or in the least not improve results. However, the fact that it generally improved the result after those points were removed shows that under the right circumstances some processing of the constructed sessions could provide additional information.

6.1 Burstiness as a summarizing metric

The burstiness measure is a two-dimensional measure so to draw conclusions it is necessary to look at the dimensions individually.

The memory variable, which quantifies the arrival pattern of actions, captures information that previously was discarded. Thus it has the potential to provide additional information in some mining scenarios. How useful it will be in practice will be a question for the future. It captures the user behavior quite well but perhaps it does not describe a session well. Another measure was discussed to address this where a session would be divided up into x chunks of equal size and then store what percentage of all the actions happens within each chunk. This measure stayed in the idea stage, but might be worthwhile investigating at some point.

The bursty variable is calculable from the mean and standard deviation of all arrival times, so it does not in fact contain any additional information if these two are present. It is a normalization of the standard deviation which otherwise does not contribute much without combining it the mean. This method of normalization was tested in another application and proved useful.

6.2 Significance of the HMM:s

The Markov models did not generate any patterns of great interest while the HMM:s generated interesting results, therefor the Markov models were dropped quite early during the data exploration.

The trouble with understanding HMM:s is that it is usually not clear what the hidden states represent. No absolute truth probably exists due to many factors contributing to the actions of users rather than one as is the assumption of the HMM. It can show an underlying trend, or give some idea of the largest factor.

Despite some similarity between the states in the HMM for the pages viewed, especially one and two, an assumption that these represent different use cases makes sense. State zero would be where users are actively searching for music and either stop there or click on an artist or an album. In state one a user clicks through different collections and play lists in order to find music. State two seems to be a session where a user is playing music from a known collection and mostly using now playing to control the music. Finally state three appears to be a session where a user visits some of the many features available to find new music, or just exploring the application in general.

The HMM for the features used gives a slightly different picture. This could be due to the fact that the data has slightly higher resolution, for instance there are more options than just "play list". Also it is possible to see such patterns that despite that artist and album pages are often accessed together it is common for users to choose one of these as the source of their play-backs. It makes sense that users looking for a specific album would pass through the artist page. If a user started playing from radio or their own play list it also shows that they are very likely to continue from those sources. Furthermore it is possible to see that some contexts seem to be highly related such as listening to Spotify made play lists or top charts or other users play lists. The other patterns that appear might be attributed to a lack of data and would need further investigation.

When having that the transition from hidden state to hidden state is unlikely with these kinds of patterns of output it is possible to talk about a change is user behavior when they appear to transition from state to state. Thus these models are suitable to use in determining where a user potentially switches what task they have in mind.

6.2.1 Potential as a summarizing statistic

Summarizing the pages viewed and features used with the general browsing or listening pattern could be very beneficial to understand higher level user behavior. It would, however, be necessary to further identify usage patterns and verify that the summary is fairly correct. The fact that the HMM models behaved the way they did seem to indicate that some general patterns could be extracted. As seen the users can also switch between different patterns during one session and including the spread between different patterns would probably be beneficial to include.

Using such a summary is probably more beneficial than for instance summarizing with which pages were viewed and how many times each was viewed.

6.3 Conclusion

A further investigation of which points should be used to generate sessions should be conducted. With the lack of a better model of which passive streams should be regarded as passive actions it appears best to assume that passive streams are not actions.

It will be necessary to test this new approach with real world mining applications and see if it is a consistent improvement. It is very possible that more advanced models are able to compensate for the small changes made

to the sessions by utilizing more metrics. However, having more accurately constructed sessions can be viewed as a goal in itself.

Transaction identification also shows some promise to generate more correct results. It would be necessary to create a method which efficiently identifies a number of different transactions in a session. This might be too much work for a small increase in performance since preliminary findings show a very low amount of sessions that clearly exhibit patterns of two or more clearly separable behaviors. At some point if sessions are utilized to a greater extent it might make sense to spend that effort in an attempt to perfect the understanding of sessions.

How to summarize a session turned out to be a more complex question than anticipated. It is possible to argue that any metric that captures any data not previously included in the summary has potential to contribute to the mining process in the future. The approach of including as many as possible is thus a good practice. It is possible to exclude a few metrics for instance the MM:s did not provide a good spread of resulting values and different browsing patterns could generate the same end result, both which are undesirable qualities of a metric. Performing a soundness check on metrics that are added could be in order to not store irrelevant data. In the same spirit a counter could be added to the different fields where employees keep track of how often a specific metric has contributed to a mining process and after a while it will become clear which metrics are a waste of space.

6.4 Future work

6.4.1 Utilizing correct session data

If it were possible to collect a large amount of data where the start and end, or breaking point, of sessions is known it would be possible to train a ML model to identify these points and perhaps create more well defined sessions. Such a model could potentially provide a better basis for mining especially if it were possible to separate users using the same account.

It is not certain that such a model would generate significantly better results but comparing to the ground truth is good practice to fully understand how well a model functions. It would also be possible to compare the resulting sessionization of different heuristics to the ground truth and compare how well they capture reality.

6.4.2 Adaptive model

Another possible approach that is worthwhile investigating is to train a ML model that splits raw data into sessions, utilizing the accuracy of the predictions performed on top of those sessions as feedback in order to create sessions optimal for those predictions. This is a very computationally costly approach as the feedback is quite lacking in extra information and a trial and error approach would be needed which is slow and unreliable to find optimal models. Also applying a ML model to the transaction identification might generate better results.

6.4.3 Further investigation of summarizing metrics

It is always possible to add more metrics to summarize a session. Further work could be put in to develop these. For example in addition to the burstiness a "user attention" measurement could be added. In fact it was investigated in another thesis and it showed some promising results. It was a somewhat crude model, which assumed that users attention peaked around actions and decayed until the next action, but it still generated some promising results. With some studies on how user attention actually correlates to the users actions might give an even better model which generates more information than only having when and in what fashion actions are performed.

The natural process to create additional metrics is to perform manual evaluation of which parts of information is lost when summarizing could be conducted. Then with this information in hand it could be reasoned how such features could be summarized. For instance in which fashion actions arrive was not previously captured but the metric burstiness keeps track of that property.

6.4.4 Formalizing the validity of summarizing metrics

Setting up a test for summarizing metrics is difficult due to the fact that a metric might not be useful in all applications, but it might provide additional information in some cases. Thus in order to exclude a metric a very rigorous set of tests would be necessary. Instead it might be more viable to perform a sanity check on added metrics, which in practice would mean to check that they have a good spread over the different sessions and to verify that two different behaviors can not generate the same summary.

Then as proposed above some kind of system could be implemented to see which summary metrics are actually used, and let the practical work be

the evaluation factor.

6.4.5 Identifying and generalizing browsing and listening patterns

HMM:s are probably not the optimal way to understand the different browsing behaviors of users, and a more relevant method such as frequent item sets or similar could be used to get more clear patterns. It would be interesting to have an accurate model, however, to both understand where user behavior changes and to utilize as a summary of the pattern in a session. How to accurately identify and summarize these is a large project, which is out of the scope of this report.

Bibliography

[1] M. Hilbert and P. López, "The World's Technological Capacity to Store, Communicate, and Compute Information," *Science 332, 60*, 2011.

[2] The Spotify Team, "About Spotify." `https://press.spotify.com/us/about/`, 2016. [Online; accessed 29-June-2016].

[3] M. J. Carey, S. Ceri, P. Bernstein, U. Dayal, C. Faloutsos, J. C. Freytag, G. Gardarin, W. Jonker, V. Krishnamurthy, M. a. Neimat, P. Valduriez, G. Weikum, K. Y. Whang, J. Widom, and B. Liu, "Web Data Mining," pp. 449–483, 2007.

[4] P. K. Srivastava, Mitali; Garg, Rakhi; Mishra, "Preprocessing Techniques in Web Usage Mining : A Survey," *International Journal of Computer Applications*, vol. 97, no. 18, 2014.

[5] J. Zhang and a.a. Ghorbani, "The reconstruction of user sessions from a server log using improved time-oriented heuristics," *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.*, 2004.

[6] P. E. Roman, J. D. Velasquez, and R. F. Dell, "Web User Session Reconstruction Using Integer Programming *," *IEEE/ACM International Conference on Web Intelligence and Intelligent Agent*, no. February 2016, pp. 2–5, 2008.

[7] P. E. Román, R. F. Dell, J. D. Velásquez, and P. S. Loyola, "Identifying user sessions from web server logs with integer programming," *Intelligent Data Analysis*, vol. 18, no. 1, pp. 43–61, 2014.

[8] C. Shahabi and F. Banaei-Kashani, "Efficient and Anonymous Web-Usage Mining for Web Personalization," *INFORMS Journal on Computing*, vol. 15, no. 2, pp. 123–147, 2003.

[9] Y. Li and B. Q. Feng, "The construction of transactions for web usage mining," *Proceedings of the 2009 International Conference on Computational Intelligence and Natural Computing, CINC 2009*, no. 1, pp. 121–124, 2009.

[10] A.-L. Barabási, "The origin of bursts and heavy tails in human dynamics," *Nature*, vol. 435, no. May, 2005.

[11] K.-I. Goh and A.-L. Barabási, "Burstiness and Memory in Complex Systems," *ArXiv*, vol. 0, no. 1, p. 4, 2006.