The complexity of modern software systems has had the unintended consequence of overwhelming users with large amounts of information, reducing their ability to effectively use these systems for their intended purposes. This phenomenon is known as the information overload problem, and is remedied by personalization. The literature provides several useful algorithms, methods, and techniques that may be used to personalize existing software systems. However, each approach has specific input data requirements. Without extensive prior knowledge of the literature, knowing which approach or procedure to employ for a given system is rarely obvious. A framework that could ease the selection of a suitable method by non-technical individuals for any given software system would be of immense value to those seeking to enable personalization. This thesis will present a framework to assist in the selection of appropriate techniques for the personalization of software systems based on the characteristics of available data.
I’d like to thank my advisors, Dr. Daniel Gillis and Dr. Judi McCuaig for their support and guidance over the course of the last two years. I would never have been able to create this document without you. You taught me to think in a research-oriented manner, to apply myself to my work, and to persevere through the most difficult periods. Dan, thank you for the direction and the multiple opportunities you’ve provided for me and School of Computer Science projects you got me involved with. These experiences have taught me a multitude of skills that have had a very positive impact on my life. Judi, thank you for your enthusiasm towards all things Computer Science. It is because of your passion for this domain and your teaching back in CIS*1500 that I chose to pursue a degree in this field. Thank you both!

Thank you to my parents, Mario and Dianne for always believing in me and continually encouraging me to work harder towards my goals.

We acknowledge that the University of Guelph resides on the ancestral lands of the Attawandaron people and the treaty lands and territory of the Mississaugas of the Credit. We recognize the significance of the Dish with One Spoon Covenant to this land and offer our respect to our Anishinaabe, Haudenosaunee and Métis neighbours as we strive to strengthen our relationships with them.
# Contents

Abstract ii

Acknowledgements iii

1 Introduction 1

1.1 Overview of Personalization 3

1.2 Thesis Statement 6

1.3 Document Outline 7

2 Categories of Personalization, Prominent Trends, and Exemplary Methods From The Literature 8

2.1 Terms and Definitions 9

2.2 Goals of Personalization 11

2.2.1 Personalization of Content 11

2.2.2 Personalization of Structure 12

2.3 Methods of Personalization 14

2.3.1 Domain-Based Filtering 14

2.3.2 Collaborative Filtering 15

2.3.3 Hybrid Filtering 16

2.4 Data Requirements for Personalization 17

2.4.1 User Interest 18
2.6.3 Personalization of Structure via Collaborative Filtering
In The Literature ................................. 36

2.6.4 The Clustering With Ensuing Classification Method .......................... 39

2.6.5 Mining Sequential and Non-Sequential Patterns ............................. 42

2.6.6 Personalization of Structure via Domain-Based Filtering ................ 48

2.6.7 Implementing Personalization of Structure via Domain-Based Filtering ............................. 48

2.6.8 Personalization of Structure via Domain-Based Filtering
In The Literature ................................. 50

2.6.9 Personalization of Content via Collaborative Filtering ...................... 54

2.6.10 Implementing Personalization of Content via Collaborative Filtering ............................. 54

2.6.11 Personalization of Content via Collaborative Filtering In The Literature ............................. 56

2.6.12 Personalization of Content via Domain-Based Filtering ...................... 58

2.6.13 Implementing Personalization of Content via Domain-Based Filtering ............................. 59

2.6.14 Personalization of Content via Domain-Based Filtering
In The Literature ................................. 60

2.7 Concluding Remarks ................................. 63

3 The Framework ................................. 66

3.0.1 System Design Constraints ........................................ 67

3.0.2 System Architecture Constraints ........................................ 68

3.1 Determining The Most Suitable Approach To Personalization For a Given System ........................................ 69

3.2 Personalization Possible Based on Characteristics of Available Data ........................................ 71
Chapter 1

Introduction

In the modern computing landscape, personalization of software systems is vital to providing users with information that pertains to their interests and informational needs, and is critical in helping users overcome information overload (Weinmann, Schneider, & Robra-Bissantz, 2013; Fernando, Du, & Ashman, 2014). The creation and perpetuation of software representations of users, user interests, or the informational needs of users is an essential task in providing personalization in any type of software system (Peter Brusilovsky, 2007; White & Chen, 2009; Ricci, Rokach, & Shapira, 2011). The common element of nearly all contemporary software systems providing some measure of personalization of content or structure to users is the presence of an underlying representation of those users and their perceived individual informational needs (Ricci et al., 2011). These representations of users are commonly referred to as user models, and are composed of many types of data pertaining to users of the software system, or usage data collected as those users utilize and interact with the software system (Frias-Martinez & Liu, 2006; Liana Razmerita & Maedche, 2003; P. B. Ahn Jae-wook & Han, 2015).
Due to the increasing complexity of present-day software systems, especially those that are web-based, users may often find themselves overwhelmed with exposure to too much information (Micarelli, Gasparetti, Sciarrone, & Gauch, 2007; Ricci et al., 2011). This problem, formally known as the information overload problem, has manifested itself in online search systems (Fernando et al., 2014; Steichen, Ashman, & Wade, 2012), news providers (Liu & Pedersen, 2010), websites and web applications (Brusilovsky, 2007; Holub & Bielikova, 2010; Barla, 2011), and software systems providing social media services (Gomez-Rodriguez, Gummadi, & Schoelkopf, 2014; Bozdag & Timmermans, 2001).

The solution to the information overload problem is not to reduce the total amount of information available to system users and consumers of data, but instead to provide personalization of either the content or the structure of software systems such that users are instead primarily exposed to the information that is deemed most relevant to their individual interests or informational needs (Bozdag & Timmermans, 2001; Brusilovsky, 2007; Weinmann et al., 2013; Ricci et al., 2011). There are many techniques, methods, and algorithms that are used to accomplish these goals via personalization. However, knowing precisely which specific technique or method to use is a daunting task for individuals who are not subject matter experts (Ricci et al., 2011). Non-subject matter experts would likely not be aware of what type of personalization is even possible for a given system based on the characteristics of the data that system produces and makes available (Adomavicius & Tuzhilin, 2005; Mobasher, 2007). Techniques which have received the most widespread support in the literature – along with required input data – will be presented in Chapter Two of this thesis.
Simplifying the process through which a method is selected to enable personalization of a software system based on the characteristics of available data would be beneficial to parties interested in reducing the extent to which users of a system are afflicted by the information overload problem. This endeavor requires first that the characteristics of available data be determined to identify which method, technique, or algorithm is best suited to be the mechanism of action through which personalization may be achieved. Finally, the types of personalization possible with the chosen technique or algorithm may be established, from which one that best suits the needs of a particular software system can be identified. These steps will be discussed in more detail in the following sections.

1.1 Overview of Personalization

Personalization is the deliberate act of filtering out information deemed to be less relevant to users of a software system and presenting them with the information most likely to satisfy their individual informational needs (Ricci et al., 2011).

\[ U = (p_1, p_2, p_3...p_n) \]

Figure 1.1: Tracking a single user’s usage of a hypothetical website. Dimensions \( p_i \in U \) are associated with a given page on a website, and the corresponding values represent whether or not the user visited the page during a session. \( n \) is the number of pages on the site. In this simple example, once the vector is populated with data pertaining to a given user’s usage of the site, each dimension will have a boolean value of true or false, and give insight into which pages the user may be interested in.

Personalization of a user’s experience with a software system can take many different forms. Many of the researchers in the surveyed literature personalize
the manner through which websites display information to users based on their browsing behaviours and patterns, such as the time they spent on pages relative to other users of the site and the order in which they view pages on the site (Peter Brusilovsky, 2007; Holub & Bielikova, 2010). Several researchers chose to represent the browsing behaviour of users on websites in similar formats – as presented in Figure 1.1 – in which the dimensions within the vector are associated with pages on the site, and the value at a given dimension is a boolean true if the user visited the page during a session, and false otherwise (Mobasher, 2007; Holub & Bielikova, 2010). Despite the fact that knowing common browsing patterns may be useful in achieving personalization for a given software system, no standard method of representing browsing patterns exists.

The user model presented in Figure 1.2 is a set of vectors, with each vector representing a single item within the system in which the user has displayed interest. In this implementation, resources are news articles in a news recommender system. Vectors representing articles are extracted via a Term Frequency-Inverse Document Frequency (TF-IDF), a technique that will be explored in more depth in section 2.5.3 of Chapter Two.

\[ U = \{a_1, a_2, a_3\} \]
\[ a_1 \in U = (0.81, 0.15, 0.44...t_n) \]
\[ a_2 \in U = (0.01, 0.18, 0.72...t_n) \]
\[ a_3 \in U = (0.12, 0.36, 0.21...t_n) \]

Figure 1.2: A user model, \( U \), is composed of vectors \( a_1 \), \( a_2 \), and \( a_3 \). The vectors are created by using TF-IDF to extract the relative importance of words in news articles a user has displayed interest in. These are vector space model representations of the documents, and make the task of finding similar documents more feasible.
While personalization improves the user experience and provides value to these systems by reducing the negative effects of the information overload problem, just two examples of instances of user models make clear that no single method for representing users exists in the surveyed literature. As will be presented in Chapter Two, an abundance of methods, techniques, and algorithms are used to manipulate data in many formats to achieve several types of personalization across numerous classes of software systems. Determining the characteristics of available data, identifying the most appropriate method to achieve personalization, and subsequently ascertaining that the selected method of personalization is the most effective for a given software system are daunting tasks that require in-depth knowledge of user models and personalization.

It is apparent that personalization of the information to which users are exposed is of vital importance to users of modern software systems; it is also clear that it is difficult to determine what type of personalization can be achieved based on the set of available user and usage data (Mobasher, 2007; Kelly & Belkin, 2004). Even more difficult is determining which techniques should be used to attain such personalization (Mobasher, 2007; Ricci et al., 2011).

Thus far we have explored personalization as a tool that can be used to improve the experience of users of a software system and reduce their exposure to information overload. However, personalization can also be used as an instrument through which marketing strategies can be applied in commercial scenarios, such as e-commerce. Whether the primary intention of personalization is to reduce information overload for users of a software system or to increase revenue for the corporation that owns, develops, and maintains the system is a dimension that is not explored in this thesis.
For individuals who are already familiar with personalization as well as the techniques and methods from the literature that can be used to effectively enable it, determining which method to use to personalize an existing software system would likely be of relative ease. However, for those who are not already experienced with these techniques and the different data requirements each of these methods possesses, selecting an appropriate technique can be very challenging (Mirone, 2011; Kwon & Kim, 2012). Frameworks have been developed to analyze software systems providing personalization from very technical perspectives, but these are not of significant value to individuals who are not already highly experienced with personalization techniques and existing methods from the literature (Kouki, Fakhraei, Foulds, Eirinaki, & Getoor, 2015; Pazzani, 1999). A tool that could enable non-technical individuals or those that are not well versed in the fields of user modelling, Statistics, or Computer Science to make high-level decisions pertaining to how any given software system might enable personalization could be useful to those who do not have technical expertise in these fields (Sunikka & Bragge, 2008; Mirone, 2011). A framework such as this could prove to be of value in determining what types of personalization are possible for a given system based on the characteristics of available data (Sunikka & Bragge, 2008; Kouki et al., 2015; Pazzani, 1999).

1.2 Thesis Statement

This thesis will introduce a framework that will assist parties seeking to personalize a software system to select an appropriate technique based on the characteristics of available user or system usage data.

The framework is intended to be primarily used by people who do not have a strong background in Computer Science, Statistics, or user modelling. People
taking advantage of the framework will be able to make informed decisions on what user modelling techniques, statistical methods, and algorithms to use to personalize user experiences within a wide variety of software systems based on the characteristics of available user and usage data. Additionally, users of the framework could rest assured that the decisions they make are supported by the most current academic research.

1.3 Document Outline

The subsequent chapters of this thesis will elaborate on the ideas and concepts introduced in this chapter. To establish the current state of the field and examine trends, case studies of quintessential and exemplary software systems enabling personalization across a wide variety of domains will be presented in Chapter Two. Chapter Two will also identify and examine the primary categories and types of personalization that have been observed in the literature, which will serve as the constituents used to create the framework. From discovered trends in the literature, Chapter Three of this thesis will present the framework that will help people make intelligent choices on what algorithms, methods, and user modelling techniques to use to enable personalization in a wide range of software systems in an effort to combat the information overload problem. These choices will be based on the characteristics of available user and usage data, and the similarities they bear to data utilized by systems in the surveyed literature. Chapter Four will apply the framework to a fictional but naturalistic data set comprised of server logs representing usage data of a website, which closely resembles the type of usage data contemporary websites collect on user browsing patterns and behaviour. Chapter Five will offer concluding remarks and discuss future work.
Chapter 2

Categories of Personalization, Prominent Trends, and Exemplary Methods From The Literature

Chapter One of this thesis introduced the role of personalization within modern software systems, and demonstrated its utility and advantages. The goal of this chapter is twofold. The first goal is to present major types of personalization found in the literature in order to discover trends in the types of data that are used as input and to unearth which techniques, algorithms, and methods are typically used in attaining different types of personalization. Several archetypal and exemplary implementations of many types of software systems providing personalization will be presented. For each implementation, the type of data most commonly required as input, the methods/algorithms used to enable personalization, the type of personalization achieved by the system will be the main points of focus. The second goal of this chapter is to discover what similarities and differences exist between these software systems and the methods, algorithms and input data they use to provide personalization. This
is done in an effort to better understand and subsequently allow for the categorization of such systems among the plethora of software systems from the literature that offer personalization in one form or another.

2.1 Terms and Definitions

This section will provide definitions for terms useful in discussing the personalization of software systems that will be used throughout this document.

In the context of this thesis, a resource is a discrete item, or source of data within a software system that is consumed by a user that can be represented by some nominal identifier. The concept of a resource is universal in the surveyed literature, though no single term is used from one system to another (Ricci et al., 2011). Some examples of resources within the context of several types of software systems are presented below:

1. Web pages within a website or web application (Holub & Bielikova, 2010; Mobasher, 2007),
2. News articles within a news recommender system (Billsus & Pazzani, 1999; Liu & Pedersen, 2010),
3. Documents or websites indexed by a search engine (e. a. Beel Joeran, 2013a), and
4. Songs in a music provider or streaming service (Kordumova, Kostadinovska, Barbieri, Pronk, & Korst, 2010; Johnson, 2014).

To better understand the use of personalization, we explore the literature under two themes: method of personalization, and goal of personalization. Method of personalization concerns itself with the processes and techniques used to determine what information is to be presented to users such that it
will most accurately satisfy their informational needs. We consider the situation where the literature can be classified under this theme as implementing \textbf{domain-based} filtering or \textbf{collaborative} filtering (Ricci et al., 2011; e. a. Beel Joeran, 2013b). Domain-based filtering is the presentation of information to individual users based on how they have previously responded to similar information. Collaborative filtering is the presentation of information to individual users based on how similar they are to other users of the same software system (Ricci et al., 2011; Mobasher, 2007).

Goal of personalization informs us about the manner in which a particular software system delivers information to its users. Similar to method of personalization, we have two classifications under the goal of personalization theme. Specifically, \textbf{personalization of content} concerns itself with providing users with the most relevant resources within a system where there is little to no structure or composition linking those resources (Ricci et al., 2011). The priority of \textbf{personalization of structure} is providing users the opportunity to navigate from resource to resource within a software system where resources are linked to each other as quickly and efficiently as possible in order to satisfy their informational needs (Razmerita, Nabeth, & Kirchner, 2012; Weinmann et al., 2013).

Providing personalization to users of any software system is ultimately a prediction of what information is most likely to satisfy the informational needs of a given user (Mobasher, 2007; Romero, Ventura, Zafra, & De Bra, 2009). Depending on the method of personalization, such a task can be accomplished by determining which system resources that user has previously displayed interest in or how similar the interests of that user are to other users of the system, by using domain-based filtering or collaborative filtering, respectfully.
There may arise an ambiguity of terms when referring to the current user of the system for whom personalization must be provided and other users of the system (to whom the user actively using the software is compared in systems employing collaborative filtering). When discussing implementations of such systems, many researchers refer to the user that is actively using the system and for whom personalization is currently being provided as the system’s active user (Romero et al., 2009; Khribi, Jemni, & Nasraoui, 2008). To avoid any ambiguity, this thesis will follow suit.

In the following sections, we will explore these two themes (goal of personalization and method of personalization), each classification, and the intersection of each classification. For each intersection, we will investigate and analyze software systems from the literature that can be classified into that intersection, in addition to examining the typical input data, algorithms, and methods used by those systems to provide personalization.

2.2 Goals of Personalization

Personalization of content and personalization of structure have been identified as the two primary categories of personalization that can be used to mitigate the effects of the information overload problem presented in Chapter One of this thesis by a wide array of researchers (Brusilovsky, 2007; Razmerita et al., 2012; Koliás, Koliás, Anagnostopoulos, Kambourakis, & Kayafas, 2008; Germanakos et al., 2009). This section will present more details on both.

2.2.1 Personalization of Content

Personalization of content embodies the personalization of information that is displayed to a given user depending on their informational needs, while not
affecting the navigational experience of users as they interact with system resources (Weinmann et al., 2013; Brusilovsky, 1996). An example of a system providing this type of personalization is Spotify™, as users are recommended new songs that are not explicitly linked to previously played songs (Johnson, 2014). Personalizing the information displayed to users based on the information comprising their user model is an effective manner of assisting users to find relevant information with less effort, and thus satisfying their informational needs while being minimally affected by information overload (Brusilovsky & Maybury, 2002; Weinmann et al., 2013; Razmerita et al., 2012). Such systems typically present different resources to different users of the same system, basing what information is presented on the inferred or explicitly stated preferences and interests of those users, without altering their navigational experience. The most ubiquitous and well known examples include Netflix™, Google News™, and online targeted advertisements (Zhou, Wilkinson, Schreiber, & Pan, 2008; Razmerita et al., 2012; Pazzani & Billsus, 2007).

### 2.2.2 Personalization of Structure

The manipulation of the structure of a given software system to help users navigate linked resources (such as pages on a website) more effectively, and thus reach informational goals more quickly and with more ease constitutes personalization of structure (Brusilovsky, 1996; Peter Brusilovsky, 2007; Weinmann et al., 2013). Personalization of structure is primarily achieved via adaptive navigation support (Peter Brusilovsky, 2007; Tsandilas et al., 2003). Adaptive navigation support aims to alter the manner through which users explore the structure of a given software system and assist users in navigating system resources more effectively. The aim is to satisfy informational needs as effortlessly as possible (Ricci et al., 2011). Systems personalizing in this manner
are typically websites or web applications (Brusilovsky, 2007, 1996). With appropriate data, and using the correct techniques, a given website may assist in personalizing a user’s navigation through pages on the site based on the browsing behaviour of previous users such that the information on the site can be accessed more effectively (Brusilovsky, 2007).

According to Brusilovsky, in his seminal work *The Adaptive Web: Methods and Strategies of Web Personalization* (Brusilovsky, 2007), adaptive navigation support may be achieved using any of the following techniques:

**Link Ordering**  Link ordering is the sorting of existing links to other resources in the system, sorted by how relevant the system deems them to be a given user, based on what information in contained within their user model (Brusilovsky, 2007; Baumeister, Knapp, Koch, & Zhang, 2005; Al Qudah, Cristea, Bazdarevic, Al-Saqqa, & Al-Sayyad, 2015). Link ordering reorders all links such that they appear to the user in the order that they are deemed to be most relevant, as opposed to just having an unsorted group of recommended pages (Alshammari, Anane, & Hendley, 2014).

**Link Hiding**  Link hiding refers to the filtering of non-relevant information or material to given users by hiding links to pages that are deemed irrelevant to those users (Brusilovsky, 2007; Raufi, Ferati, Zenuni, Ajdari, & Ismaili, 2015; Alshammari et al., 2014). This approach may aid to reduce the user’s cognitive load, and serves to reduce the amount of non-pertinent information users are exposed to, based on their knowledge, interactions with the system, or informational goals (Brusilovsky, 2007; Alshammari et al., 2014).

**Link Generation**  Link generation refers to the process of adding completely new links to other resources within the system (Raufi et al., 2015; Brusilovsky,
2007). Brusilovsky refers to Amazon’s online marketplace recommender system as a prominent example of link generation, and explains that this is perhaps the most popular method of providing adaptive navigation support (Brusilovsky, 2007).

2.3 Methods of Personalization

Collaborative filtering and domain-based filtering have emerged as two of the most widely used methods of personalizing user experiences in recommender systems and other software systems offering personalized content or structure (e. a. Beel Joeran, 2013b; R. B. Koren Yehuda & Volinsky, 2001; Adomavicius & Tuzhilin, 2005). Both methods provide effective means to accurate and practical recommendations. This section aims to discuss the conceptual and implementation differences between the two approaches and their respective strengths and weaknesses.

2.3.1 Domain-Based Filtering

Domain-based filtering recommends information and resources to users by assigning levels of interest or relevance to particular resources within a software system and recording their features, and recommending yet unseen resources with similar features to that user (Steichen et al., 2012; Liu & Pedersen, 2010; Balabanovic & Shoham, 1997). For this approach to function, a standard set of features representing each resource in the software system must be available (Pazzani & Billsus, 2007).

The underlying concept which underpins domain-based filtering is that a given user’s interests or informational needs may be inferred from past interests and informational needs (Bilenko, 2008). As such, a software system should
be able to recommend new resources for any given user based on their levels of interest in resources they have interacted with in the past (Gaudioso & Boticario, 2002), (Bilenko, 2008). This interest level metric may be inferred from past behaviours or habits or even explicit ratings pertaining to system resources (Pazzani, Muramatsu, Billsus, et al., 1996; Mobasher, 2007; Holub & Bielikova, 2010). More information on determining user interest levels in system resources may be found in section 2.4.1 of this chapter. Many statistical methods and algorithms can be employed to achieve this domain-based filtering and several detailed examples are presented in sections 2.6.6 and 2.6.12 of this chapter.

Domain-based recommender systems have an extremely wide range of applications, including the recommendation of scholarly and education materials and information, (e. a. Beel Joeran, 2013b), news article suggestions based on user interests (Billsus & Pazzani, 1999; Pazzani & Billsus, 2007), recommending web pages to users (Holub & Bielikova, 2010; Pazzani & Billsus, 2007), and providing adaptive navigation support in hypermedia systems (Peter Brusilovsky, 2007; Pazzani et al., 1996).

2.3.2 Collaborative Filtering

Collaborative filtering allows for personalization by analyzing how similar the data pertaining to the interests of an active user is to that of other users, and providing recommendations of resources to the active user to which the most similar other users responded positively (e. a. Beel Joeran, 2013b; e. a. Goldberg David, 1992; Brusilovsky, 2007). The term was first coined in 1992 by David Goldberg (e. a. Goldberg David, 1992). The approach functions based on the assumption that users whose past informational needs are similar will
also have similar informational needs in the future (e. a. Beel Joeran, 2013b; e. a. Goldberg Ken, 2001; Ricci et al., 2011).

Many statistical methods to gauge the similarity between representations of user interests have been explored in the literature, including classification algorithms such as $k$-Nearest Neighbours and statistical techniques like the Pearson correlation coefficient (Holub & Bielikova, 2010; Mobasher, Dai, Luo, & Nakagawa, 2002b). Several examples of software systems making use of the collaborative approach to provide personalization are presented in this chapter. Namely, sections 2.6.1 and 2.6.9 present strategies employing the collaborative approach to a wide variety of software systems.

2.3.3 Hybrid Filtering

Hybrid filtering is a method that makes use of both domain-based and collaborative filtering (Choi, Yoo, Kim, & Suh, 2012; Bobadilla, Ortega, Hernando, & Bernal, 2012). Hybrid filtering can be useful in eliminating the cold-start problem sometimes experienced by systems implementing collaborative filtering, whereby the system has difficulty providing meaningful personalization when resources have not yet been viewed by many users (Basiri, Shakery, Moshiri, & Hayat, 2010; Bobadilla et al., 2012; Choi et al., 2012). However, implementations utilizing this approach to filtering are relatively uncommon in comparison to those that choose to use either domain-based or collaborative filtering exclusively (H. J. Ahn, 2008; Hu & Pu, 2011). This lack of popularity is primarily due to the inherent increased complexity and implementation challenges when compared to the more commonly used implementations which utilize either domain-based filtering or collaborative filtering (Su & Khoshgoftaar, 2009; H. J. Ahn, 2008). Due to the underrepresentation of systems
employing hybrid filtering as a method of personalization in the literature, the
framework will focus on domain-based and collaborative filtering.

2.4 Data Requirements for Personalization

To achieve any type of personalization of a user’s experience with a software
system, data pertaining to their knowledge of the system, their perceived infor-
mational needs, or usage data representing their interactions with the system
must be available (Frias-Martinez & Liu, 2006; Ricci et al., 2011; Mobasher,
2007). Such data can be present in several formats, and it is necessary to be
able to determine the characteristics of that data to determine how that data
might be used to enable personalization, and also in order to determine which
methods and algorithms may be appropriately used to accomplish such a goal.

For example, simply knowing that a given user of a website is interested in
page a of a particular website, but is not interested in page b of that same
website are important details in providing a personalized experience for that
user, but the manner in which such data is represented is vital in determining
how to achieve a personalized experience for that user. If the extent to which
users are interested in pages on the website can be represented by some piece
of normalized interval data (where these values are maintained in a vector,
and dimensions within the vector are associated with pages on the site), then
users sharing similar interests may be discovered using a wide variety of tech-
niques (Holub & Bielikova, 2010; Mobasher, 2007). For example, the cosine
distance between the vectors representing the interests of users may be used
to quantify the most similar other users, and subsequently recommend pages
other similar users have found useful in satisfying their informational needs
(Holub & Bielikova, 2010). While such an example may work in this particular
situation, without the appropriate data available it would be of no use in providing personalization to a software system (Frias-Martinez & Liu, 2006; Ricci et al., 2011). Knowing specific details about the characteristics of the available data are of utmost importance in selecting an appropriate method, technique or algorithm through which to attain a given form of personalization (Belk, Germanakos, Andreou, & Samaras, 2015; Peter Brusilovsky, 2007). Table 2.1 presents several approaches to personalization from the literature along with associated data requirements and the techniques used to enable personalization.

This section will present the most common data requirements for personalization from the surveyed literature; the following section will present several associated methods used with this data.

2.4.1 User Interest

In order to achieve personalization of any variety, some form of feedback is required in order to gauge what resources individual users are interested in (Kordumova et al., 2010; Billsus & Pazzani, 1999).

Two primary methods of gathering data pertaining to user interest in system resources exist. Explicit feedback, which as the name suggests, entails gathering data from users simply by asking them how interested they are in a particular resource (Frias-Martinez & Liu, 2006; Billsus & Pazzani, 1999). Alternatively, implicit feedback can infer user interest in system resources without requiring users of that system to explicitly state which resources are of interest (Koutri, Avouris, & Daskalaki, 2005; Mobasher, 2007). A variety of observable
Objective | Input Data | Method | Personalization | Researchers |
---|---|---|---|---|
| | Vector of nominal data | Naive Bayes classifier | | Billsus, 1999 |
Collaborative user classification | Set of nominal resource identifiers | $k$NN classification | | Mobasher, 2007 |

Table 2.1: Approaches from the literature used to achieve personalization in several software systems offering personalization to users, and associated input data.

User actions (which will be discussed below) can be used to determine the extent to which users are engaged and interested in individual system resources (Mobasher, 2007; Holub & Bielikova, 2010). Implicit feedback is significantly more popular in the surveyed literature; it is convenient and less intrusive from a user’s perspective, and users are often unwilling to provide explicit feedback (Koutri et al., 2005; Mobasher, 2007; Qiu & Cho, 2006).

### 2.4.1.1 Common Implicit Interest Indicators From The Literature

1. **Time on Page**

   Positive integer, interval data. Typically measured in seconds, as the difference between two timestamps of resource access times (Gündüz &
Özsu, 2003; Kramar & Bielikova, 2012). This is by far the most common method used in the surveyed literature (Bahadori, Harounabadi, & Sadeghzadeh, 2013; Holub & Bielikova, 2010; Kim, Atluri, Bieber, Adam, & Yesha, 2004; Khribi et al., 2008). To mitigate false positives in resource interest, it is common practice to discard any value above a certain threshold as the user may have left their device unattended while viewing the resource (García, Romero, Ventura, & De Castro, 2009).

2. **Resource Access**

Nominal value representing whether or not a given resource was accessed by a user (e. a. Beel Joeran, 2013b; Mobasher, 2007; Bandari, Xiang, & Leskovec, 2017).

3. **Interactions With Resources**

Representing number of scrolling actions, clicks on page, uses of the clipboard (Holub & Bielikova, 2010). Positive integer, interval data (Holub & Bielikova, 2010)

### 2.4.1.2 Common Uses of Explicit Feedback

To a lesser extent, some researchers in the surveyed literature have used explicit feedback in order to determine which system resources given users are interested in (Billsus & Pazzani, 1999; Kordumova et al., 2010). One example can be found in section 2.6.1 of Chapter Two. As an example of explicit feedback, the feedback categories users of the news recommendation system implemented by (Billsus & Pazzani, 1999) are presented with the following options:

1. **Interesting**
2. **Not interesting**
3. **I already know this**
4. **Tell me more**
Ordinal data may also be used as explicit feedback in order to quantify user interest in system resources. The music recommendation system with xStream by (Kordumova et al., 2010) uses a 5 point explicit rating scale, items are then paired with the implicit feedback from that same item, and a Naive Bayes Classifier is able to classify new items into either like or dislike categories.

In order to qualify as explicit nominal feedback, the data must be either nominal or interval data, the semantics of which are rooted in explicit user feedback relating to a given identifiable system resource. Again, the use of explicit feedback is significantly less common than the use of implicit feedback, due primarily to the inconvenience it poses to users (Koutri et al., 2005; Mobasher, 2007; Qiu & Cho, 2006)

2.4.2 Session Level Data

Sessions are sequences of interactions during which a particular user accesses system resources during an individual visit to a website, web application or other software system (Mobasher, 2007; Koutri et al., 2005).

To qualify as session level data, a set of resources each user accessed during their session must be available. For web-based systems, server logs might be a collection of HTTP requests to the server, where each request contains at least the nominal URI requested and a session identifier (Sampath, Mihaylov, Souter, & Pollock, 2004; Bianco, Mardente, Mellia, Munafo, & Muscariello, 2009). Additional common data associated with web logs may include the timestamp of each request, user locale, and the client’s IP information (from which geolocation may be inferred) (Koutri et al., 2005). Sessions ordered by timestamps are referred to as click-streams (Koutri et al., 2005). Session level data is used as input for several data capture mechanisms presented in
the following section, including association rules mining and sequence patterns mining. If this data is not explicitly available, sessionization heuristics can sometimes be employed to extract session level data from server logs. This will also be addressed in section 2.5.1 of this chapter.

2.4.3 Feature Vectors

Feature vectors are the single most common method of representing resources in systems employing domain-based filtering, while vectors of interest values are commonly used to represent users in systems making use of collaborative filtering from the surveyed literature (Ricci et al., 2011; Mobasher, 2007; Billsus & Pazzani, 1999). When representing resources, these feature vectors are composed of interval, boolean, or nominal data points, with each dimension within the vector as relating to a particular property of that resource (Ricci et al., 2011). An example of such a property is the frequency of a given term in the natural language associated with that resource relative to other resources (Ricci et al., 2011; Billsus & Pazzani, 1999).

2.4.4 Vectors of Interest Values

When vectors are used to represent users in collaborative filtering systems, they often do so by tracking the perceived interest users have in system resources (Ricci et al., 2011). In these systems, each dimension within an interest vector is associated with a specific system resource (Holub & Bielikova, 2010). A vector of interest values is generated for each user, with individual values within these vectors being the product of the time a user has spent interacting with that dimensions associated resources or some other user interest metric (Mobasher, 2007; Holub & Bielikova, 2010). Techniques for generating these vectors will be explored in more detail in the coming sections.
2.4.5 Natural Language

As it is one of the most abundant forms of data, several software systems that provide personalization do so by using natural language as input data (Ricci et al., 2011). Resources composed of natural language – like news articles – can be used as the input for applications such as news recommender systems, despite having no associated feature vectors (Billsus & Pazzani, 1999; Liu & Pedersen, 2010). These systems do so by processing natural language and quantifying the contexts of the natural language so that different bodies of text may be compared for similarity (e.g. Beel Joeran, 2013a). These techniques will be explored in more detail in section 2.5, Data Capture Mechanisms.

2.4.6 Vector Space Models

Vector space model representations of documents of natural language are vectors in which dimensions are associated with terms in the corpus to which the document belongs (e.g. Beel Joeran, 2016). Values within the vector are a function of how important the associated term is in the represented document (Billsus & Pazzani, 1999). Term Frequency-Inverse Document Frequency is one of the most common methods for obtaining vector space model representations of documents in the surveyed literature, as will be explored in section 2.5.3 of this chapter. Creating such representations of documents of natural language is important because it allows for quantifiable comparison of the natural language within documents (Ricci et al., 2011; Billsus & Pazzani, 1999).

2.4.7 Sequential Patterns

A sequential pattern is a sequence of resources that are accessed by an active user of a software system in chronological sequence (Mobasher, 2007). Using techniques that will be outlined in section 2.5, they may be mined to discover
which navigational patterns are most commonly used within a given software system. Two primary categories of sequence patterns exist: contiguous and open (Krištofič & Bieliková, 2005; Mobasher, 2007). In order to match a contiguous pattern, a candidate pattern must match a $n$ items in the contiguous pattern, all in a row. On the other hand, to match an open pattern, the candidate pattern must simply match $n$ items in the open pattern, but they do not need to be contiguous. To illustrate, an example:

Let $P = \{p_1, p_2, ..., p_5\}$ be the set of webpages on a given site.

If we consider a candidate subset $\{p_1, p_3, p_5\}$, the candidate subset would:

1. **Contiguous Patterns**
   - Match $\{p_1, p_3, p_5\}$
   - Not match $\{p_1, p_2, p_3, p_5\}$
   - Not match $\{p_1, p_5, p_3\}$

2. **Open Patterns**
   - Match $\{p_1, p_3, p_5\}$
   - Match $\{p_1, p_2, p_3, p_5\}$
   - Not Match $\{p_1, p_5, p_3\}$

Thus, sequential patterns are the chronological order in which an active user has accessed resources within a system.

### 2.4.8 General Navigational Patterns

General navigational patterns represent the least restrictive type of pattern matching, as they do not consider the chronological order in which resources are accessed, unlike open sequence and closed sequence patterns (Mobasher, 2007).
2.5 Data Capture and Analysis Mechanisms

In order to understand the contexts in which the data requirements presented in the previous section are used, this section will present several techniques from the literature used to create or format the data types presented in the previous section.

2.5.1 Sessionization Heuristics

From a data collection and analysis point of view regarding web usage logs, the ideal situation is to have a unique identifier representing which session each request belongs to, obtained from data stored within cookies, web tokens, or some other method that can be used to uniquely identify users making HTTP requests during a single session. If such an identifier is not available, sessionization heuristics may be employed.

Sessionization is a procedure which entails determining which HTTP requests belong to the same user during one given use of the site (Mobasher, 2007). The process of sessionization refers to the organization of server logs and site browsing behaviour into a more structured, cataloged format (Mobasher, 2007). The input required is a set of server logs, where each log contains the following information that may be used to uniquely identify users and sessions (Román, Dell, Velásquez, & Loyola, 2014; Berendt, Mobasher, Nakagawa, & Spiliopoulou, 2002):

1. IP Address of client
2. User-agent (browser)
3. Page requested
4. Timestamp of the request
The time-based heuristic to sessionization has been widely adopted by many researchers, due to its simplicity and effectiveness in identifying user sessions (Román et al., 2014; Berendt et al., 2002). The heuristic functions as follows: in order to be considered to be part of a single session, sets of HTTP requests must originate from the same IP address, and must have been initiated by the same user-agent (Berendt et al., 2002; Swapna, Guptha, & Geetha, 2017; Ansari, Babu, Ahmed, & Azeem, 2011). This is used in order to uniquely identify users.

To determine whether two subsequent requests from the same user belong to the same session, a set time interval, $t$ (measured in seconds), must not have elapsed between the two requests. If the amount of seconds that has elapsed between two requests is greater than $t$, then the requests are said to belong to two different sessions (Román et al., 2014; Berendt et al., 2002). The session lasts as long as there is a request made by the user within the indicated interval (Berendt et al., 2002). Many time intervals have been used, ranging from 300 (Román et al., 2014) to 1800 (Berendt et al., 2002; Li, Sun, Dunham, & Xiao, 2003) seconds. The specific value chosen is left to the discretion of the user of the framework and the owner of the data.

### 2.5.2 Similarity Measures

In domain-based filtering, resources are compared to each other and measured for similarity; such a metric is known as a *similarity measure* (Y. Koren & Bell, 2015). Within the literature, the most common method for representing resources is to represent them as vectors (Desrosiers & Karypis, 2011; R. B. Koren Yehuda & Volinsky, 2001; Billsus & Pazzani, 1999). In the surveyed literature, there exist two primary statistical methods most commonly
used to find the similarity between vectors of data associated with or representing resources within software systems (Desrosiers & Karypis, 2011; Mobasher, Dai, Luo, & Nakagawa, 2002a). These two methods are the Pearson correlation coefficient and cosine similarity (Nguyen, 2014; Vozalis & Margaritis, 2003; Billsus & Pazzani, 1999). These methods owe their widespread use in part due to their simplicity and ease of implementation (Nguyen, 2014). Additionally, cosine similarity has been demonstrated to be more effective in determining similarity between resources when dealing with vectors of high dimensionality (Ertöz, Steinbach, & Kumar, 2003).

While these two methods are the most popular in the surveyed literature, others exist, including Jaccard distance, Manhattan distance, and Euclidean distance. Users of the framework may use whichever method most closely fits their needs.

2.5.2.1 Pearson Correlation Coefficient

The Pearson correlation coefficient is a statistical method used to determine the linear correlation between two sets of data points (Benesty, Chen, Huang, & Cohen, 2009). The technique can be used to find the similarity (correlation) between two vectors of data points, and has been used by several researchers in the surveyed literature (Holub & Bielikova, 2010; Baltrunas & Ricci, 2009; Benesty et al., 2009).

2.5.2.2 Cosine Similarity

The cosine similarity of two vectors has been widely used in the field of information retrieval to determine how similar two text documents are to each other (Singhal, 2001). Weighted vectors of nominal data are extracted from the bodies of text using techniques such as term frequency, or TF-IDF, and
the cosine similarity between these vectors is used to judge how similar the contents of the documents are to each other (Singhal, 2001).

2.5.3 Vector Formatting Techniques

Several algorithms exist that may be used in order to produce vectors from raw text, representative of the terms and contents of that text (Van Meteren & Van Someren, 2000). These algorithms each yield a vector of either boolean or interval values that may be used to quantify the differences between documents or bodies of text, and are rooted in the field of Information Retrieval (Billsus & Pazzani, 1999). Suitable candidate resources for the application of these methods must contain bodies of natural language or text.

2.5.4 Term Frequency

Term frequency may be used to construct vectors of interval data representing the importance of a word to a given document of text (Abel, 2011). The output of this process is a vector of interval data, which may be used to represent the document.
2.5.5 Term Presence

The presence of a word in a resource indicates that the associated dimension within the vector should contain a value of 1. Otherwise, if the word is not present the value at that dimension is 0 (Pazzani et al., 1996; Billsus & Pazzani, 1999). This approach is less popular than the others in the surveyed literature.

2.5.6 Term Frequency-Inverse Document Frequency

Term Frequency - Inverse Document Frequency is often used to evaluate the importance of terms within a corpus of text, and is the most popular in the surveyed literature (Pazzani et al., 1996). The method functions by considering the frequency with which any given term occurs in a given document, but the frequency value is counterbalanced, or offset by the term’s frequency within the set of all documents in a collection (Van Meteren & Van Someren, 2000). This ensures that common words are not treated as overly important, but that words appearing in fewer documents are adequately represented (Van Meteren & Van Someren, 2000). The method is a standard technique for creating vectors of data that represent text documents so that they may be compared for similarity (Bailey, El-Beltagy, & Hall, 2001; Billsus & Pazzani, 1999; Van Meteren & Van Someren, 2000).

The term frequency for term \( t \) is simply the number of times the term occurs within the document. The Inverse Document Frequency of that term is the log of the total number of documents divided by the number of documents in the collection that contain term \( t \) (Van Meteren & Van Someren, 2000).

\[
IDF(t) = \log \frac{n}{n_t}
\]
where $n$ is the total number of documents in the collection, and $n_t$ is the number of documents containing that term. TF-IDF is then calculated as:

$$TF-IDF(t) = tf(t) \cdot \log \frac{n}{n_t}$$

This formula is used to calculate values for each term in the document, and the output is a vector of interval data that may be used to represent the document (Van Meteren & Van Someren, 2000; e. a. Beel Joeran, 2013b; J.-w. Ahn & Brusilovsky, 2009; Billsus & Pazzani, 1999). Several researchers in the surveyed literature have employed this technique, many examples of which are presented in this chapter.

2.5.7 Association Rules Mining

Using this technique, sets of pages accessed most frequently during any given session will be revealed (Mobasher, 2007). Input data required for this approach is a set of sessions, where sessions are unordered sets of pageviews, with pageviews being identified by some nominal identifier such as a URI.

When mining association rules from a set of sessions, there are two primary goals (Mobasher, 2007):

1. To discover frequent itemsets within the data
2. To discover association rules

Discovering Frequent Itemsets To discover frequent itemsets – resources frequently accessed together – the Apriori algorithm, or algorithms derived from Apriori are often employed (Kumar & Rukmani, 2010; Mobasher, 2007). The algorithm functions by determining or sessions within a data set which appear at least $n$ times (Agrawal, Srikant, et al., 1994; Mobasher, 2007), where
is a \textit{minimum support threshold} value. The size of the itemsets is expanded with each iteration, adding new frequent itemsets of larger and larger size until the \textit{minimum support threshold} requirement can no longer be met (Agrawal et al., 1994; Mobasher, 2007). Support for a given itemset is the frequency with which it appears in the set of sessions (Mobasher, 2007).

**Discovering Association Rules** Out of the set of discovered frequent itemsets from the previous step, it must then be determined how often two of those itemsets appear within the same session (Agrawal et al., 1994; Mobasher, 2007).

### 2.5.8 Sequence Patterns Mining

Mining of sequential patterns is similar to association rules mining in the sense that both approaches attempt to find resources that are often accessed together during the same session (Mobasher, 2007). However, while association rules simply determine subsets of resources that are likely to be accessed during a given session, sequence pattern mining introduces a chronological restraint on access patterns, in the sense that the order in which users access resources is used to determine pattern creation (Mobasher, 2007). As with association rules, two of the most common metrics that may be used to quantify the extent to which a given sequential pattern is true are \textit{support} and \textit{confidence} (Mobasher, 2007).

Again, the \textit{support} of a pattern determines the frequency with which a given frequent pattern occurs relative to all sessions (Mobasher, 2007). For example, let $S$ be the set of all sessions. Let $F$ be the set of frequent patterns, as discovered by a frequent itemset mining algorithm:

$$F = \{F_1, F_2, \ldots, F_n\}$$

The \textit{support} for a given frequent pattern $F_i$ is (Mobasher, 2007):
support($F_i$) = $\frac{|\{s \in S : F_i \text{ isSubsequenceOf } s\}|}{|S|}$

where $S$ is the set of all sessions. The confidence of a rule that $X$ implies $Y$ can be stated as (Mobasher, 2007):

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X \text{concat} Y)}{\text{support}(X)}$$

Both Apriori and PrefixSpan may be used, but PrefixSpan is more efficient in terms of both computational and space complexity, and both yield the same result (Kumar & Rukmani, 2010). However, Apriori is a conceptually simpler algorithm and is more easily implemented (Kumar & Rukmani, 2010).

Now that data requirements for personalization and common data capture methods from the literature have been defined, we may explore the roles they play in providing several types of personalization.

### 2.6 Intersections of Themes and the Four Approaches to Personalization

The intersections of the two themes through which we explore the literature – method of personalization and goal of personalization – can be used to identify four approaches to personalization, as presented in Figure 2.1. These are:

1. Personalization of structure via collaborative filtering
2. Personalization of structure via domain-based filtering
3. Personalization of content via collaborative filtering
4. Personalization of content via domain-based filtering
Each of these four approaches provides personalization to software systems in an attempt to reduce the negative effects of information overload on users of those systems, but do so in different manners, and utilize different procedures and methods. The following four sections will present several software systems from the surveyed literature for each of these four approaches, along with their respective input data requirements, associated methods and algorithms used to enable personalization, and the specific details pertaining to the type of personalization they provide.
2.6.1 Personalization of Structure via Collaborative Filtering

Software systems utilizing collaborative filtering depend on user feedback, either implicit or explicit, in order to determine whether an active user is interested in a given resource (Holub & Bielikova, 2010; Mobasher, 2007). When personalizing structure, systems employing collaborative filtering ascertain what other similar users have found to be useful or popular strategies of navigating the structure of a software system and provide adaptive navigation support to the active user (Mobasher, 2007; Weinmann et al., 2013). For example, providing personalized recommendations of which page on a site to visit next might alter the sequence in which a user accesses pages on a site, leading them to information that satisfies their informational needs more effectively (Mobasher, 2007).

2.6.2 Implementing Personalization of Structure via Collaborative Filtering

From a high-level perspective, all systems from the surveyed literature providing personalization via collaborative filtering achieve personalization in three stages (Desrosiers & Karypis, 2011; Holub & Bielikova, 2010; Mobasher, 2007).

1. Determine interests and build user models
2. Find similar users
3. Recommend resources

Starting with these three high level steps, we will further analyze systems from the literature providing personalization of structure via collaborative filtering.
Determine Interests and Build User Models  Because systems in this category personalize structure, most implementations from the surveyed literature are web based, and as such use session data pertaining to which resources users viewed as input data (Mobasher, 2007; Holub & Bielikova, 2010; AlMurtadha, Sulaiman, Mustapha, & Udzir, 2010). These implementations typically contain an offline and an online component, but there are exceptions, as described in section 2.6.3 of this chapter. The offline component of these systems analyzes data from users who have previously used the software system, and format the data such that the online component can efficiently provide personalization of the system to an active user in real time (Desrosiers & Karypis, 2011; Mobasher, 2007).

A popular method used by researchers in the literature to analyze the data generated by past users of the system is to discover the most frequent navigational paths and patterns users have engaged in (Mobasher, 2007; AlMurtadha et al., 2010; Spiliopoulou & Faulstich, 1998). This may be accomplished by using association rules to determine which resources in the system users typically access together, or by clustering users based on their usage data, and subsequently creating aggregate user models that represent archetypal users of a given system based on navigational patterns discovered in session level data (Mobasher, 2007). These aggregate user models are tracked as vectors of interest values pertaining to their interest in system resources (Holub & Bielikova, 2010; Desrosiers & Karypis, 2011).

Find Similar Users  Due to the collaborative nature of this approach, it is important to track how the active user makes use of the software system in order to determine how they compare to users who have used the system in the past. Implementations from the literature track the active user as they
navigate the software system, and format this data in the same manner as was done for the offline component (Desrosiers & Karypis, 2011). A common implementation from the literature is to create a vector of \( n \) dimensions, where \( n \) is equivalent to the number of resources in the system (Mobasher, 2007; Holub & Bielikova, 2010; Desrosiers & Karypis, 2011). This representation of the active user’s interest can then be compared to the aggregate user models generated by the offline component using some statistical measure such as the pearson correlation coefficient or cosine similarity, or use a classifier to find the \( k \) most similar users (Holub & Bielikova, 2010; Mobasher, 2007).

**Recommend Resources** Collaborative filtering functions based on the assumption that users who have shared similar informational needs in the past will also share these needs in the future (Desrosiers & Karypis, 2011). As such, the resources most likely to be of interest to the active user – those most likely to satisfy their informational needs – can be inferred from the interests of users similar to them (Holub & Bielikova, 2010). The resources viewed by past users of the system whose navigational behaviours are most similar to the active user become candidates for recommendation, by which they may be used to implement link generation or an equivalent adaptive navigation support technique to assist the active user in navigating from one resource to the next (Holub & Bielikova, 2010).

### 2.6.3 Personalization of Structure via Collaborative Filtering

**In The Literature**

This section will present various archetypal software systems from the literature which provide personalization of structure via collaborative filtering, along with their associated methods, algorithms used to enable personalization, and the type of data these systems typically use as input.
One system from the surveyed literature presents collaborative adaptive navigation that is able to improve the navigational experience of users on a website by providing personalized suggestions of which pages on the site to browse next based on which pages they have viewed during their session, and their behaviours while viewing those pages (Holub & Bielikova, 2010). An interest value representing user interest a given web page is generated from this user data. This interest value for any given page is a function of several actions the user takes while browsing the site relative to the average level of actions for that page across all users. For both time on page and number of scrolling events, a user is determined to be interested in the webpage if their values are higher than the average user (Holub & Bielikova, 2010). Unfortunately, the researchers do not go into great detail about how this interest value is calculated. This system uses implicit feedback such as the time the active user spends viewing resources as its input, and creates interest vectors for each user. The resulting interest value is a normalized interval value between 0 and 1 (Holub & Bielikova, 2010).

In this implementation, as an active user browses the site during a given session, a vector of these interest values will be constructed (Holub & Bielikova, 2010). In order to provide adaptive navigation support, the Pearson correlation coefficient is used as a similarity measure in a collaborative approach to find the user who has the most similar vector of interest values (Holub & Bielikova, 2010). Pages viewed by the other user that have not been visited by the active user are then recommended via link generation, and provide adaptive navigation support (Holub & Bielikova, 2010). Note that while this approach provides effective personalization, it has the disadvantage of scaling linearly with the total number of visitors to the site.
Other researchers in the literature offer exceptionally similar personalization via comparable methods, but employ an offline component to create clusters of access patterns in order to aid their system scale with great efficiency (AlMurtadha et al., 2010; Mobasher et al., 2002a). Systems employing such offline components to carry out the most computationally intensive tasks are typically categorized as using web usage mining techniques (Romero et al., 2009).

Web usage mining techniques are more scalable because they reduce the amount of data that must be processed to provide recommendations in real time (Romero et al., 2009; Mobasher et al., 2002b). Each of these techniques are comprised of an offline and an online component (Mobasher, 2007). This is in contrast to more conventional systems – such as the implementation discussed at the beginning of this section – which simply compare the active user to all past users of the system in order to determine page recommendation (Van Meteren & Van Someren, 2000; Holub & Bielikova, 2010). In systems employing web usage mining techniques, the offline component is typically responsible for mining through session level data and extracting the most common usage patterns or navigational paths on a given site, which are subsequently used as input to the online component (Mobasher et al., 2002b). This is done in order to reduce the computational workload of the online component, which must function in real time when providing recommendations to users.

Several researchers identify three web usage mining techniques that are widely used and have received significant support in personalizing web systems, and which represent a large portion systems that can be classified as providing personalization of structure via collaborative filtering in the surveyed literature.
1. Clustering with ensuing classification
2. Sequential patterns mining
3. Association rules mining

Trends found in the surveyed literature indicate that these methods typically use session level data as input (Mobasher, 2007). The nature of the offline and online components, as well as the most popular methods for web usage mining will be explored in the following section (Koutri et al., 2005). We begin with the clustering with ensuing classification method.

### 2.6.4 The Clustering With Ensuing Classification Method

Many researchers implement personalization capable of providing adaptive navigation support on any given website by determining the most common navigational patterns used to explore the structure of the site by past users, and then determine whether or not the active user’s browsing behaviour is similar to that of past users (AlMurtadha et al., 2010; Mobasher et al., 2002a). Common navigational paths are stored in structures referred to as aggregate user models (Mobasher et al., 2002a). An offline component handles clustering of session data and creation of aggregate user models (a time consuming, computationally expensive operation), while the online component handles real-time recommendations derived from the aggregate user models and the active user’s current navigational behaviour, inferred from session data (AlMurtadha et al., 2010; Kim et al., 2004; Bandari et al., 2017; Romero et al., 2009; Khribi et al., 2008).
Session level data are typically converted to vectors to track the level of interest each user has in pages on the site, with dimensions of these vectors associated with pages on the site and the values at dimensions are a function of user interest in the page (AlMurtadha et al., 2010; Mobasher, 2007). Alternatively, the values stored within these vectors may be a function of an implicit interest indicator, such as time spent viewing or interacting with a page or may simply track whether or not the user has visited a page or resource, in which case their values are restricted to boolean true or false (Holub & Bielikova, 2010; Mobasher et al., 2002a, 2002b; AlMurtadha et al., 2010). With most implementations from the literature, the underlying assumption and most common theme regarding user interest is that the more time a user spends viewing or interacting with a given resource, the more interested they are in that resource (Holub & Bielikova, 2010; Mobasher, 2007; Ricci et al., 2011).

**Offline Component** Sessions that log interactions with the software systems are processed such that they are represented as vectors of values (either boolean values, or interval, as described above), where the weight assigned to a given dimension in a vector is a function of the amount of time a user spent on the page, or simply whether or not the user visited the page (AlMurtadha et al., 2010). From this set of vectors, $n$ clusters representing common navigational patterns are created by applying a clustering algorithm to the data (Mobasher, 2007). A popular choice is the $k$-means clustering algorithm, because the number of clusters is always controlled, so the scalability of the online component is always constant (AlMurtadha et al., 2010; Mobasher, 2007).

Aggregate user models are created by taking the mean vector of all vectors belonging to a given cluster, this is the cluster centroid (AlMurtadha et al., 2010; Mobasher, 2007). Therefore, the number of aggregate user models is
proportional to the number of clusters created. The format of such an aggregate user model is presented below.

\[ AGM = p_1, p_2, ..., p_n | p \in P, weight(p) \geq weight_{\text{min}} \]

where \( P \) is the set of all pages on the website, and \( weight_{\text{min}} \) is a minimum weight, ensuring that pages with very low levels of interests within a cluster are not included in the aggregate user model (AlMurtadha et al., 2010). If the average time the page was viewed was below the threshold, the weight for that page is set to zero (AlMurtadha et al., 2010). No consensus in the surveyed literature exists for such a threshold, as the amount of time that is typically spent viewing any given resource varies greatly from one software system to the next.

**Online Component** When the active user browses the site, their session data is processed such that it is in the same vector format as those created by the offline component (Baraglia & Silvestri, 2007; AlMurtadha et al., 2010). The closest aggregate user model to the active user’s user model is essentially discovered using \( k \)-Nearest Neighbours classification, with \( k = 1 \) (Mobasher, 2007). In order to determine how to personalize the structure of the site for this active user, the aggregate user model that is the single nearest neighbour of the vector created for the active is calculated via cosine similarity (AlMurtadha et al., 2010; Mobasher, 2007). Pages contained in the aggregate user model that have not yet been viewed by the active user are recommended via an adaptive navigation support technique (AlMurtadha et al., 2010). The manner in which the active user is exposed to the structure of the site is personalized to their perceived informational needs by collaboratively determining how they resemble past users of the site. Another web usage mining technique common
in the surveyed literature that shares this goal is pattern mining, discussed below.

### 2.6.5 Mining Sequential and Non-Sequential Patterns

Another method to determine which resources are typically accessed together is to analyze session level data of past users of the system, and use association rule mining to determine the most commonly visited pages during sessions (Mobasher et al., 2002b). Studying usage patterns and uncovering frequent navigational patterns is a viable method of determining how users have utilized a software system, and may in turn be used to provide personalization to future users (Koutri et al., 2005).

This method functions with an offline component to prepare the data and discover patterns, similar to the clustering with ensuing classification method (Mobasher et al., 2002a; AlMurtadha et al., 2010; Bandari et al., 2017). The offline component of the system handles data preparation and pattern discovery, while the online component performs recommendations in real time to an active user in order to provide adaptive navigation support (Mobasher et al., 2002b).

**Offline Component** In the surveyed literature, three approaches to pattern discovery and the analysis of user navigational patterns on websites are used:

1. **Frequent Itemsets** - Resources often viewed together during a session, order is irrelevant.
2. **Sequential Patterns** - Resources often viewed in sequence, order is relevant.
3. **Sequential Contiguous Patterns** - Resources often viewed together during a session, order is relevant and patterns must be contiguous.
In order to discover frequent itemsets in the browsing patterns of past users, the only information required is a set of resources accessed by a given user during a given interaction with the system (Mobasher et al., 2002b). This is because frequent itemsets do not consider chronological order of access when discovered resources frequently accessed together (Mobasher, 2007). However, session level data must be available to perform sequential pattern discovery, as a timestamp associated with each request must be available, as the chronological order in which resources were accessed is required to determine frequent navigational paths through a site’s resources.

**Frequent Itemsets** Frequent itemsets (pages often visited together during a single session) are discovered through the use of association rule mining (Mobasher et al., 2002b). Such techniques reveal what pages on a given website are most typically accessed together during a single session (Baraglia & Silvestri, 2007). A frequent itemset mining algorithm, such as the Apriori Algorithm, is typically been used to perform this task (Mobasher, 2007).

**Sequential Patterns** For sequential patterns and contiguous sequential patterns, the Apriori Algorithm is modified so that it instead discovers frequent subsequences instead of frequent itemsets (Mobasher et al., 2002b). Alternatively, sequential pattern mining algorithms such as PrefixSpan or CloSpan may be used (Romero et al., 2009; Han, Pei, & Yan, 2005). Because the goal of this approach is to mine and analyze sequences of resource requests, the chronological order in which pages are accessed is considered (Han et al., 2005). An implementation from the literature uses a special form of tree known as a prefix tree to store these frequent sequences in which each node in the tree is representative of a particular sequence of page accesses (Mobasher et al., 2002b). Storing frequent patterns in this manner allows for very efficient searches of
the mined sequential patterns and enables rapid adaptive navigation support by the system’s online component (Mobasher et al., 2002b). An example is provided in Figure 2.2.

**The Online Component** In real time, as the active user browses the site, the tree produced by the offline component is searched for a frequent itemset or a sequential pattern (depending on what the offline component generated) matching the active user’s session (Mobasher et al., 2002b). If one is found, the children of that node will contain the same frequent itemset with an additional page, which is recommended to the user (Baraglia & Silvestri, 2007). This implementation allows the online component to execute in $O(|s|)$ time, where $|s|$ is the cardinality of the active user’s session (Mobasher et al., 2002b).

Figure 2.2 presents an example illustrating this concept using sequential patterns. The offline component has discovered several frequent sequential patterns through which users most commonly navigate the resources on the site from the session level data of past users. It has created a prefix tree using these frequent sequential patterns. Each of the nodes in the tree represents a single sequential pattern, and its children (if any) are supersets of their parent nodes, along with one extra resource. The hypothetical website being examined has five pages.

1. /home
2. /personal
3. /accounts
4. /estimates
5. /trends
For this example, let us say that the active user has so far visited the pages /home, /accounts, in that order. Using the identifiers of those pages, the active session is \( \{1,3\} \). Performing depth first search on the prefix tree, it can be found that there is a sequential pattern \( \{1,3\} \), stored in node \( n \). The value stored at \( n \)’s child node is the sequential pattern \( \{1,3,4\} \). Thus, the suffix of this child node contains the identifier of the page to be recommended (Mobasher et al., 2002b). The page with identifier 4, /estimates, would therefore be recommended to the user. In the event there are multiple children to the node \( n \) in the prefix tree containing the active user’s session, the association rule in which the node \( n \) is the antecedent and \( n \)’s children are the consequent that has the most confidence can be used to determine which page to recommend.
to the active user (Mobasher et al., 2002b). In the context of association rules, confidence pertains to the extent to which the antecedent implies the consequent (Mobasher et al., 2002b).

It is by using this approach that (Mobasher et al., 2002b) are able to provide adaptive navigation support. Similar approaches are employed by many other researchers in the literature (Kim et al., 2004; Bandari et al., 2017; Romero et al., 2009; Khribi et al., 2008; García et al., 2009).

<table>
<thead>
<tr>
<th>Input Data Type</th>
<th>Input Data Source and Semantics</th>
<th>Method</th>
<th>Output of Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering - Almurtadha, 2010 and Mobasher, 2002</td>
<td>Set of vectors of interest values (sessions)</td>
<td>K-means</td>
<td>Aggregate user models</td>
</tr>
<tr>
<td>Sequence Patterns - Mobasher, 2007</td>
<td>Set of vectors of interest values (sessions)</td>
<td>Pattern discovery algorithm for mining frequent patterns</td>
<td>Set of frequent navigational paths</td>
</tr>
<tr>
<td>Sequence Patterns - Mobasher, 2007</td>
<td>Set of sessions, with request timestamps</td>
<td>Pattern discovery algorithm for mining frequent patterns</td>
<td>Set of frequently accessed itemsets</td>
</tr>
</tbody>
</table>

Table 2.3: Offline components of several web usage mining systems from the literature. Associated input data required, algorithm used to discover frequent patterns or aggregate user models, and output of stage are presented.

In each implementation of a web usage mining technique, the offline component uses a set of sessions pertaining to usage of a site as input, and produces as output a set of aggregate user models, providing insight pertaining to which
resources users typically access together during a given session, or sets of frequent navigational paths on the site (Han et al., 2005; Bandari et al., 2017; Romero et al., 2009; Mobasher, 2007). The online component of these systems subsequently either determines which aggregate user model is most similar to the active user’s session, or determines whether or not the active user’s current session resembles a frequently browsed path or a set of pages on the site that are frequently accessed together during a single session (Baraglia & Silvestri, 2007). These specialized methods used to enable personalization of structure via collaborative filtering each ensure that the operations performed by the online component are not computationally expensive, and as such they may be reliably be performed in real time (Baraglia & Silvestri, 2007; Mobasher, 2007).

Due the the navigational nature of the web (to which personalization of structure applies), and the fact that users of any given site all communicate with a remote server, usage data from all users of the system is often available (satisfying the requirements of collaborative filtering) to systems providing personalization to websites and web applications (Baraglia & Silvestri, 2007). As a consequence of this – and as was presented in this section – there are a wide variety of software systems from the literature that implement personalization of structure via collaborative filtering. The following section will present the second of the four approaches to personalization: personalization of structure via domain-based filtering.
2.6.6 Personalization of Structure via Domain-Based Filtering

While underrepresented in the literature in comparison to personalization of structure via collaborative filtering, personalization of structure via domain-based filtering is still a viable manner through which the structure of websites, web applications, and other generalized recommender systems can be personalized (Van Meteren & Van Someren, 2000). This underrepresentation is likely due to the fact that software systems seeking to personalize structure are mostly web-based, where usage data is readily available, making collaborative filtering a popular approach (Baraglia & Silvestri, 2007; Linden & York, 2003).

2.6.7 Implementing Personalization of Structure via Domain-Based Filtering

Personalization of structure via domain-based filtering is possible if usage data from the system’s users is available, and the user’s interactions with resources within the system is a navigational activity. Data required for this approach are typically feature vectors that pertain to system resources, either directly related to the resources themselves or extracted from the natural language contained in resource via a method such as TF-IDF, detailed in section 2.5.3 of this chapter. Along with these feature vectors, data representing which resources users have viewed – and optionally, how interested users are in those resources – are required. Steps and stages used across relevant implementations from the surveyed literature that are used to personalize in this manner will be presented in this section. Researchers Lops, De Gemmis, and Semeraro present three primary components or steps required to provide personalization to domain-based systems (Lops, De Gemmis, & Semeraro, 2011):
1. Content analysis
2. Profile learning
3. Filtering component

**Content Analysis** Content analysis in the surveyed literature consists of determining what standard method can be used to represent resources such that they may be meaningfully compared to one another (Billsus & Pazzani, 1999). Resources in the surveyed literature are most often represented as feature vectors, using data that is common in all resources in the system (Lops et al., 2011; Abel, 2011).

**Profile Learning** Profile learning involves tracking which system resources the active user has viewed (Abel, 2011; Pazzani et al., 1996). If the level of interest the user has in these resources is known, the profile for the active user consists of a set of tuples, containing the users interest level in the resources they have viewed, along with a feature vector representing the associated resources. In more dated implementations of personalization of structure via domain-based filtering, users were asked to explicitly rate their interest levels in each browsed webpage (Pazzani et al., 1996). This represents a major inconvenience to users, and modern systems instead use time on page to infer interest in resources (Van Meteren & Van Someren, 2000; Mobasher, 2007). The inferred level of user interest, as well as the associated feature vector representing a resource are added to their profile as the active user navigates the system and views new resources.

**Filtering Component** Since resources are represented as feature vectors, many implementations use a distance metric to determine which yet unseen resources to recommend to an active user (Billsus & Pazzani, 1999; Pazzani et al., 1996). The average of all feature vectors contained within the user
model are calculated (Pazzani et al., 1996). By discovering which yet unseen resources are most similar to the resources an active user has previously viewed or displayed interest in, the navigational experience of the user may be personalized in order to ensure that the structure of the system is optimized such that that user’s informational needs may be satisfied with the least amount of effort. Personalization is achieved using any of adaptation navigation support techniques, such as link generation, outlined in section 2.2.2 of this chapter.

2.6.8 Personalization of Structure via Domain-Based Filtering In The Literature

Several examples of systems exemplifying domain-based filtering to provide personalization of structure will be presented in this section.

Citing the increasing complexity of modern websites, researchers Van Meteren and Van Someren discuss the need for an adaptive navigation support system aimed at helping users more effectively navigate the resources provided by websites via personalization based on the contents of the site’s pages (Van Meteren & Van Someren, 2000). The researchers describe a system named PRES (Personalized Recommender System), which uses vector space model representations of webpages are compared for similarity using cosine similarity, in an approach remarkably similar to the news article recommender introduced by (Billsus & Pazzani, 1999). These systems both use natural language as input data, and use vector formatting techniques to create a standardized method of representing documents. Based on the active user’s displayed level of interest in already viewed resources, the level of interest this user might have in yet unseen pages is calculated (Van Meteren & Van Someren, 2000). Levels of interest are inferred from the amount of time a user spends viewing a given resource (Van Meteren & Van Someren, 2000). As with many other systems in
the literature, the underlying assumption regarding user interest within PRES is that the more time a user spends on a given webpage, the more interested they are in that page’s content (Van Meteren & Van Someren, 2000; Holub & Bielikova, 2010; Mobasher, 2007).

The interests of the active user are maintained within a user model which consists of a single vector that is an amalgamation of the vector space model representations of pages the user has displayed interest in (Van Meteren & Van Someren, 2000). While browsing a site, if a user spends more than a pre-defined amount of time on a webpage, the vector space model representation of the page is extracted via TF-IDF, and the active user’s user model is amended using the following equation:

$$P_t = \alpha P + \beta D$$

where $\alpha$ is a constant between 0 and 1 that reduces weightings of old interests, and $\beta$ represents the level of interest the user has shown in that resource (Van Meteren & Van Someren, 2000). Once a certain amount of seconds are spent on the website, $\beta$ is set to 1, otherwise the document’s contents are deemed to not be of interest to the user (Van Meteren & Van Someren, 2000). The researchers comment that any value could have been used for $\beta$, a boolean representation was used for simplicity’s sake (Van Meteren & Van Someren, 2000). $D$ is the current document in vector space model representation. Cosine similarity is then used to compare the user model to all other web pages in the software system, mirroring the approach used by other researchers in the surveyed literature to find similar text documents (Billsus & Pazzani, 1999; Abel, 2011). PRES provides adaptive navigation support via link generation, as was described earlier in this chapter, helping the active user navigate to the site.
resources that are most likely to be of interest (Van Meteren & Van Someren, 2000).

Other researchers construct a system to provide personalization of structure via domain-based filtering to recommend web pages that may be of interest to users, based on their past explicit ratings of web pages (Pazzani et al., 1996). A very similar implementation is provided by (Tsandilas et al., 2003). Users are asked to explicitly rate their interest levels in web pages as they browse the Internet (Pazzani et al., 1996). Available ratings include hot, lukewarm, or cold, and represent the labels for categories used by a classification algorithm in the recommendation stage (Pazzani et al., 1996). While doing so, boolean feature vectors are extracted from the natural language contents of pages on the site, and represent whether or not a word is present on a given web page (Pazzani et al., 1996). Representing documents in this manner is not a popular approach in the literature when compared to representing documents as vectors extracted via \textit{TF-IDF} (Billsus & Pazzani, 1999).

As an active user browses the web and rates viewed web pages, the vector maintaining their interests is updated, consisting of the boolean feature vectors of pages they have viewed, and their associated ratings (Pazzani et al., 1996). User interest in pages that have been viewed is gathered explicitly, by requesting the active user to rate pages as either hot, lukewarm, or cold (Pazzani et al., 1996). This nominal data point representing user interest associated with viewed pages is subsequently used to perform classification of yet unseen pages (Pazzani et al., 1996). Pages that have been linked to from the current page are prefetched, and their boolean feature vectors representing present words are created (Pazzani et al., 1996). These boolean feature vectors are then classified in order to determine how a user would classify the web page (hot, lukewarm,
or cold) (Pazzani et al., 1996). If the page is classified into the hot category, then a small icon indicating to the user that it may be of interest to them is placed next to the link to that page (Pazzani et al., 1996). Conversely, if the page is classified into the cold category, an icon suggesting the user avoid the page is presented (Pazzani et al., 1996). Depending on configuration of the system, either a Naive Bayesian classifier or the $k$-Nearest Neighbours algorithm are used to classify the boolean feature vectors representing web pages (Pazzani et al., 1996).

Each user receives personalized recommendations for web pages that are tailored to their interests (Pazzani et al., 1996). Another very similar and more recent implementation is provided by (Khribi et al., 2008).

These systems all use vectors representing system resources, typically gathered from the natural language associated with resources via $TF-IDF$ or an equivalent technique, along with some metric using to determine user interest in the resources as input (Billsus & Pazzani, 1999; Van Meteren & Van Someren, 2000). By using some statistical measure such as cosine similarity to determine the most similar vector representation of yet unseen resources, or by using a classifier to determine if the unseen resources resemble those the active user has displayed interest in, the manner through which the structure of the system can be personalized via one of the adaptive navigation support techniques previously described in this chapter can be determined.

The following section will present the trends in data requirements and methods used in software systems from the literature that provide personalization of content via collaborative filtering.
2.6.9 Personalization of Content via Collaborative Filtering

Personalization of content via collaborative filtering presents users with content that is deemed to satisfy their informational needs based on their shared preferences with other users of the same software system (Ricci et al., 2011). Details on these systems are implemented, along with prominent examples from the literature will be presented in this section.

2.6.10 Implementing Personalization of Content via Collaborative Filtering

The steps taken to enable personalization of content via collaborative filtering in the literature are remarkably similar to those used to provide personalization of structure via collaborative filtering. The primary difference lies in the fact that systems providing personalization of content do not provide adaptive navigation support, and instead provide recommendations of resources that are not related to each other. These offline components are thus often simpler and less specialized. The steps used to achieve personalization of content via collaborative filtering are:

1. Determine interests and build user models
2. Find similar users
3. Recommend resources

Determine Interests and Build User Models  Software systems employing collaborative filtering to provide personalization must track the extent to which users are interested in system resources (Ricci et al., 2011). This is typically achieved by observing user behaviours when interacting with system resources in order to determine which resources given users are interested in
Data representing an active user’s interest in a given resource can be ordinal, binary, or nominal (Desrosiers & Karypis, 2011). If using binary data, the value associated with a given resource for an active user is typically set to true if the user has viewed the resource, and false if they have not (Mobasher, 2007). Nominal data might describe actions the user has engaged in with a given resource, such as purchasing an item in an online marketplace (Desrosiers & Karypis, 2011).

In the surveyed literature, the most common approach to determining user interest in a resource is outputting an interval data point pertaining to a given resource which is a function the time spent viewing that resource (Holub & Bielikova, 2010; Mobasher, 2007).

In the surveyed literature, values stored within a user model pertaining to user interest in a given resource are thus a function of the time spent interacting with the resource, as described in section 2.4.1. This process of building a user model is useful to describe and represent in a standard format the informational needs of users (Ricci et al., 2011). In the surveyed literature, the most common method used to format user interest is to maintain a vector of interest values, as described in section 2.4.4 of this chapter (Mobasher, 2007; Ricci et al., 2011).

**Find Similar Users** Users most similar to the active user are typically determined via some similarity measure, such as the cosine similarity of two vectors of interest values (Amatriain & Pujol, 2015). In the surveyed literature, classification algorithms are a popular method for determining which users are most similar to the active user (Desrosiers & Karypis, 2011; Amatriain & Pujol, 2015). $k$-Nearest Neighbours represents the single most popular approach, owing in large part due to its simplicity and ease of implementation (Amatriain & Pujol, 2015). The most similar $k$ users can be found and resources those users were interested in can be recommended to the active user. Resources that have
been determined to be interesting by these $k$ most similar users are added to a set of resources which are candidate recommendations (Ricci et al., 2011).

**Recommend Resources** The set of candidate resources is iterated over in order to discover resources that have not yet been viewed by the active user, and those that have already been viewed by the active user are dropped from the set (Ricci et al., 2011). The remaining resources are sorted based on their interest values, and those rated as most interesting by similar users are recommended to the active user. In this manner, personalization of content is achieved.

### 2.6.11 Personalization of Content via Collaborative Filtering

**In The Literature**

This section aims to introduce the trends in characteristics of input data, methods, and algorithms used to provide personalization to software systems from the literature that are classified as personalizing content via collaborative filtering. These systems bear many similarities to systems offering personalization of structure via collaborative filtering, as presented in section 2.6.1, but differ in the fact that the navigational experience of users is not affected. This section will explore these techniques and methods, and trends in the surveyed literature.

Researchers Liu and Pedersen presents a recommender system for Google News\textsuperscript{TM} which provides suggestions of news articles that users may find interesting, based on their geophysical proximity to other users of the system as well as the an interest vector representing interest in resources they have accessed in the past, based on their resource access history (Liu & Pedersen, 2010). Researchers who have created systems providing personalized access to information have demonstrated that users in similar geographic locations
also share many informational needs (García-Crespo et al., 2009; Schedl & Schnitzer, 2014).

As with the systems presented in the two previous sections, classifiers can be used by collaborative based recommender systems to group users together based on their explicitly stated or inferred interests in system resources (Kordumova et al., 2010; Baltrunas & Ricci, 2009; Smaaberg, Shabib, & Krogstie, 2014). One such example of a software system providing personalization of content via collaborative filtering dispenses film recommendations to its users (Baltrunas & Ricci, 2009). The system uses a vector of interval data points representing the feedback rating users provided for films as input to a classification algorithm to determine which users share the most similar film interests (Baltrunas & Ricci, 2009). The researchers use \(k\)-Nearest Neighbours in order to find users who have similar rating histories in order to provide recommendations to users via the collaborative filtering approach (Baltrunas & Ricci, 2009). The Pearson correlation coefficient is used as a similarity measure to determine user to user similarity (Baltrunas & Ricci, 2009).

Another similar system from the literature, Concert Finder, gives users music recommendations (Smaaberg et al., 2014). This is accomplished by tracking the musical tastes of an active user as interval data points in an interest vector, in which each musical artist in a set is assigned a dimension, and the value at that dimension represents the number of times the active user has been to a concert featuring that artist (Smaaberg et al., 2014). The \(k\)-Nearest Neighbours algorithm in order to enable collaborative recommendations to users, using cosine similarity as a similarity measure to determine which others users are most similar to the active user (Smaaberg et al., 2014). Concert Finder recommends the artists that the \(k\) most similar users listen to the most to the
active user (Smaaberg et al., 2014). As discussed in section 2.6.1, utilizing $k$-Nearest Neighbours to personalize in this manner scales linearly with the number of users in the system, a scaling issue that could be alleviated by the use of an offline component which generates aggregate user models.

Software systems providing personalization of content via collaborative filtering in the surveyed literature are typically general recommender systems, as they simply personalize the content users are exposed to based on how similar it is to content to which they have already positively responded. As input data, these systems require the extent to which users are interested in resources the software system recommends, as described in section 2.4.1 (Baltrunas & Ricci, 2009; Smaaberg et al., 2014). The most common approach from the surveyed literature is to format user interests as vectors of interval data pertaining to user interest in system resources, and subsequently using $k$-Nearest Neighbours or a similarity measure to discover other users of the system who share interests with the active user and recommend resources the active user may not yet have discovered (Baltrunas & Ricci, 2009; Smaaberg et al., 2014).

In contrast to the collaborative filtering employed by this approach to personalization, the following section will present examples from the literature of software systems based on domain-based filtering, while still personalizing the content users are exposed to.

2.6.12 Personalization of Content via Domain-Based Filtering

Software systems personalizing content via domain-based filtering do so by finding the resources an active user has not yet viewed that are most similar to resources they have previously displayed interest in. Steps taken to implement
personalization of content via domain-based filtering, along with prominent examples of such systems from the literature will be presented in this section.

### 2.6.13 Implementing Personalization of Content via Domain-Based Filtering

As with personalization of structure via domain-based filtering, data pertaining to system resources is typically present in the form of a feature vector (Lops et al., 2011). The trends from implementations in the surveyed literature that are used to personalize in this manner will be presented below. As per Lops, De Gemmis, and Semeraro, the three primary components or steps required to provide personalization to domain-based systems are (Lops et al., 2011):

1. Content analysis
2. Profile learning
3. Filtering component

**Content Analysis** The steps taken to determine user interests in personalization of content via domain-based filtering are no different that those presented in the equivalent stage when personalizing structure via domain-based filtering. Refer to section 2.6.6 of this chapter.

**Profile Learning** As with content analysis, the actions required in this stage are identical to those taken when learning a profile for personalization of structure via domain-based filtering. Refer to section 2.6.6.

**Filtering Component** The filtering of resources is used to determine which resources yet unseen by the active user are most similar to the resources that
user has positively responded to in the past. Because resources are most commonly represented as feature vectors, the use of a similarity such as the cosine similarity of two vectors, this task is trivial (Billsus & Pazzani, 1999; Pazzani et al., 1996; Desrosiers & Karypis, 2011). However, unlike personalization of structure via domain-based filtering, systems providing personalization of content do not provide adaptive navigation support, but instead simply present the top $n$ most similar resources to the active user in the user interface of the system being personalized (Linden & York, 2003). Thus, they cannot utilize the techniques outlined in section 2.2.2 of this chapter.

### 2.6.14 Personalization of Content via Domain-Based Filtering In The Literature

Archetypal examples of domain-based recommender systems will be presented in this section, and trends in the methods for formatting system resources – often available as documents of text containing natural language, such as news articles – such that they may be compared to each other will be examined.

One system providing personalization of content via domain-based filtering from the literature is a personalized news article domain-based recommender system for Twitter™, and recommends news articles to users based on their tweeting habits (Abel, 2011). In this system, an active user’s interests are represented as a feature vector, where values within the vector are a function of how often the user tweets about a given term. The underlying assumption of this implementation is that the more often a user tweets using a given hashtag, the more they are interested in that topic (Abel, 2011). In order to determine which news items should be recommended to an active user, each news story in the system that has not yet been viewed by the user is a candidate for
recommendation, and is converted to its vector space model representation via a vector formatting technique (such as TF-IDF, as detailed in section 2.5.3) after which cosine similarity is used to compare the news stories to the active user’s term frequency vector (Abel, 2011).

A news recommender application from the literature named News Dude creates domain-based recommendations based on explicit user feedback to previously read news articles employs a similar approach to representing system resources (Billsus & Pazzani, 1999). However, in this recommender system, users are required to explicitly state whether or not they were interested in the contents of each article after it has been read. These explicit responses provide the classes for each rated news article in the training set, as described by (Van Meteren & Van Someren, 2000). News stories are represented in their vector space model representations, as vectors of interval data extracted via TF-IDF (Billsus & Pazzani, 1999). News Dude uses the k-Nearest Neighbours algorithm to classify new, yet unrated new articles into one of these classes, based on how the user has classified previously read news articles, using cosine similarity as a similarity measure to determine how similar the vectors representing different news stories are (Billsus & Pazzani, 1999).

Kordumova et al. introduce a domain-based music recommender system that uses feature vectors of several interval and nominal data points representing features associated with system resources (in this implementation, resources are songs) (Kordumova et al., 2010). Unlike the news article recommender systems presented by (Billsus & Pazzani, 1999) and (Abel, 2011), features representing system resources relate to the manner in which users engage with the resources, including the number of times the active user has listened to the song, the times they have skipped the song, and the time of day during which
the song was played (Kordumova et al., 2010). As is the case with the news recommender system using $k$-Nearest Neighbours to recommend news stories, this implementation also uses explicit feedback from users (Kordumova et al., 2010). Whether a yet unheard song should be classified to be recommended or not for an active user is determined by a Naive Bayesian classifier (Kordumova et al., 2010). This implementation is similar to the system providing adaptive navigation support presented in section 2.6.6 by (Pazzani et al., 1996), but does not provide personalization of structure.

Another domain-based recommender system from the literature is able to recommend research articles to users based on their viewing habits and interactions with scholarly articles within the Docear academic literature management system (e. a. Beel Joeran, 2013b). The system uses the natural language contained within documents as input data (e. a. Beel Joeran, 2013b). As with the previously presented software systems systems providing personalization of content via domain-based filtering, in this implementation documents are represented in vector space model format produced via $TF-IDF$ (e. a. Beel Joeran, 2013b). In this system, resources are scholarly articles in Docear (e. a. Beel Joeran, 2013b). The vector space model representations the scholarly articles the active user has saved are compared to the vector space model representations of scholarly research articles within Docear’s database by way of cosine similarity (J. Beel & Langer, 2015), as is the case with the systems presented by both (Abel, 2011) and (Billsus & Pazzani, 1999). Articles determined most similar by cosine similarity that the user has not already read are then recommended (e. a. Beel Joeran, 2013a).
The software systems from the literature that can be classified as providing personalization of content via domain-based filtering include those personalizing the scholarly articles, news articles, or songs that users of these systems are recommended. As with software systems providing personalization of structure via domain-based filtering, representing resources as feature vectors of interval data points is common (Billsus & Pazzani, 1999; Abel, 2011; e. a. Beel Joeran, 2013b). The subsequent classification of new resources based on their similarities to resources the active user has already interacted with along with that user's interest level in those resources is commonly achieved using either \( k \)-Nearest Neighbours, or a Naive Bayes classifier, which have been demonstrated to both be viable techniques for achieving personalization of content via domain-based filtering (Abel, 2011; e. a. Beel Joeran, 2013b; Billsus & Pazzani, 1999).

2.7 Concluding Remarks

This chapter introduced two themes through which we can analyze software systems from the literature that provide personalization. The first of these two themes is goal of personalization, referring what the software system personalizes. The second theme is method of personalization. Within method of personalization, software systems are classified as providing personalization via either collaborative filtering and domain-based filtering.

From these two themes, we categorize software systems into four distinct approaches to personalization: personalization of structure via collaborative filtering, personalization of structure via domain-based filtering, personalization of content via collaborative filtering, and personalization of content via
domain-based filtering. Software systems providing personalization of structure via collaborative filtering typically enable adaptive navigation support on websites based on common use cases of past users of the system. Those producing personalization of structure via domain-based filtering are used to provide adaptive navigation support to an active user on a website based on that user’s previous interactions with the system. Systems providing personalization of content via collaborative filtering provide resource recommendations based on common informational needs of users, such as a shared informational need based on geolocation. Finally, software systems personalizing content via domain-based filtering dispense recommendations of unrelated resources within the system, and are used as film recommender systems, music recommender systems, along with scholarly or news article recommender systems.

It is clear from the surveyed literature that several approaches to personalization can be employed to enable many forms of personalization. To a non-technical person or to a non-subject matter expert for the domain of personalization, knowing which type of personalization is possible based on the characteristics of available data is not evident, much less so choosing the optimal approach. A framework that analyzes the characteristics of available data could greatly simplify the decision making process used to choose the best fitting approach to personalization for individuals seeking to apply personalization techniques to a software system.

The following chapter will introduce such a framework. The goal of this framework is to guide non-subject matter experts to the appropriate approach most likely to successfully provide personalization to any given software system based on characteristics of the system that is to be personalized. The framework will subsequently aid in discovering specific methods, algorithms,
and techniques from the surveyed literature that will be most effective in providing personalization to that software system based on the characteristics of available data.
Chapter 3

The Framework

This chapter will present a framework that will aid those aiming to personalize a software system to make an informed decision regarding an optimal approach to personalization. The principle goal of the framework is to guide its users to specific methods, algorithms, and techniques from the surveyed literature that will be most effective in personalizing a given software system based on the characteristics of available data.

The framework has been divided into a set of six flowcharts, each with accompanying footnotes to explain the labels assigned to nodes in the corresponding flowchart. The framework is divided into sub-flowcharts for two main reasons. The categorization of questions associated with each of the four approaches to personalization, and an increased level of readability due to reduced clutter.

Each decision point within the framework, each process, and each claim is supported by research in the surveyed literature. Each flowchart’s node labels or footnotes will make reference to sections within Chapter One, Chapter Two, or Appendix One where examples of a given method, algorithm, or process that has been used to enable personalization within the literature may be found.
A user of this framework would be able to confidently state that their choice of technique, algorithm, or method used to provide personalization is sound within a given context, and has proven effective in the surveyed literature in a similar context. After using the framework to discover the most appropriate method to personalize a given software system, users of the framework will be able to make a statement similar to:

We may use statistical method or algorithm \( x \)

to achieve personalization of type \( y \)

based on available data with characteristics \( z \).

In practice, such a statement might more closely resemble:

We can use the \textit{k Nearest Neighbours} algorithm to
determine which users share informational needs, and

\textbf{achieve adaptive navigation support via link generation}

based on a \textbf{boolean feature vector} representing browsing behaviour during
a session.

To facilitate the process of determining the most appropriate approach to enable personalization in any scenario, system design and system architecture constraints must be examined.

3.0.1 System Design Constraints

A system design constraint may have the consequence that a given approach to personalization is not recommended by the framework. For example, a system designed to personalize content, such as Spotify\(^\text{TM}\) or one of the many news recommender systems presented in section 2.6.9 of Chapter Two, do not
by design provide personalization of structure when recommending resources to
users. This is because resources are not linked to each other, therefore the user’s
interactions with the system are not navigational. This constraint is a direct
result of system design and the nature of the software system in question. The
constraint does not negatively impact the effectiveness of personalization, but
instead simply aids to find the most relevant implementations of personalization
from the literature. Several methods, algorithms and techniques pertaining to
such software systems are specialized to provide optimal personalization given
the characteristics of available data.

In some systems, it is not clear whether or not user interactions with system
resources are navigational in nature (Brusilovsky, 2007). An example of such
an ambiguous system is Amazon’s™ online marketplace, which is essentially
a massive recommender system of largely unrelated resources, but in which
the user’s shopping experience may be viewed as an exploration to discover
the optimal resource (Brusilovsky, 2007). Determining whether or not a user’s
experience interacting with system resources is navigational in nature is left to
the discretion of the user of the framework.

3.0.2 System Architecture Constraints

The second perspective from which these systems will be viewed is that of
system architecture. The fact that the aforementioned music streaming system,
Spotify™, is internet-based, means that usage data pertaining to user interest
in resources for all users is readily available. In turn, this means that these
systems may provide personalization via collaborative filtering, which would
not be possible if the system did not have usage data pertaining to user interest
in resources available. After all, collaborative filtering is simply not possible if
the interests of an active user cannot be compared to the interests of past users,
as shown in Chapter Two. When determining which type of personalization is possible for a given software system, requiring that usage data pertaining to user interest in resources be available to measure an active user against past users presents a system architecture constraint. Systems that do not have such data available, such as those that do not communicate user interests to a server, simply cannot use collaborative filtering as a method of personalization. The framework has been constructed such that viewing the four approaches from the perspective of system design and system architecture constraints allows for a reliable selection of the most appropriate method or technique by which to implement personalization.

3.1 Determining The Most Suitable Approach To Personalization For a Given System

The four approaches to personalization can thus be ranked based on these constraints, namely their requirements of both system design and system architecture, provisioned by, respectfully:

1. Inter-resource structure or links
2. Usage data pertaining to user interest in resources, for all users

The four approaches are presented in Table 3.1, along with their associated requirements for both of these provisions concerning available input data and system structure. The approaches are ranked starting with the most restrictive.

Within the surveyed literature, software systems that are classified as sharing both goal of personalization and method of personalization often also share a variety of implementation details. Analyzing a software system for which personalization is required from the perspective of system design and system
<table>
<thead>
<tr>
<th>Approach</th>
<th>Require Resources Linked</th>
<th>Require Usage Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personalization of Structure via Collaborative Filtering</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Personalization of Structure via Domain-Based Filtering</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Personalization of Content via Collaborative Filtering</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Personalization of Content via Domain-Based Filtering</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

Table 3.1: The four approaches to personalization with associated system design and architecture requirements, ranked most to least restrictive. In giving equal weight to the system design constraint and the system architecture constraint, personalization of structure via domain-based filtering and personalization of content via collaborative filtering are equally restrictive, and their positions in this ranking are interchangeable.

Organizing the four approaches in this manner is practical because many of the methods and algorithms used to personalize systems that can be classified as less restrictive can be used to provide personalization of systems that can be classified as more restrictive, but not other way around. For example, many of the techniques and methods from the literature that represent personalization of content via domain-based filtering can be used to personalize systems that are categorized as the more restrictive personalization of structure via collaborative filtering, while the opposite is typically not possible.

To more concretely establish this idea, consider one of the approaches viewed from the perspective of system design and system architecture constraints. From this perspective, personalization of structure via collaborative filtering
is the most restrictive. It requires that both usage data pertaining to interest in system resources be available from all users (for collaborative filtering), and also requires that system resources be linked so that a user’s experience is navigational in nature (to personalize structure, not just the content users view). It is clear that personalization of structure via collaborative filtering is the most restrictive from the perspective of system and architecture requirements. Its associated methods, techniques, and implementation details are very refined and compatible with relatively fewer use cases than less restrictive approaches, as shown in Chapter Two. Determining which of the four approaches a given software system is categorized as is an important starting point in determining the most effective methods, algorithms, and implementation details for personalizing that system.

The following sections will present the flowcharts that compose the framework. These flowcharts contain more specific questions about the characteristics of available data. Based on the answers to these questions, the framework will determine the most appropriate approach to personalization for a given software system. The flowcharts contain some technical questions that pertain to the characteristics of available data, and may be updated by technical individuals to include language that is more accessible to non-technical users if desired.

3.2 Personalization Possible Based on Characteristics of Available Data

Figure 3.1 serves as the global entry point to the framework, and is a useful tool for determining not only which approach is most likely to suit a given software system, but also which subsequent flowchart should be addressed.
in order to determine the most appropriate methods to enable personalization for that system.

Figure 3.1: The Global Entrypoint flowchart, serving as the starting point to the framework, based on the system design and system architecture constraints outlined in Table 3.1.
Figure 3.2: Flowchart A. Evaluating the viability of personalization of structure via collaborative filtering based on characteristics of available data.
3.2.1 Node Labels for Chart A

Does the data set consist of session level data? Session level data is defined in section 2.4.2 of Chapter Two.

Sessionization heuristics Refer to section 2.5.1 of Chapter Two for an explanation of sessionization heuristics.
Figure 3.3: Flowchart B, Evaluating the viability of personalization of structure via domain-based filtering based on characteristics of available data.

3.2.2 Node Labels for Chart B

Interest in Resources  Are there nominal or interval data associated with system resources that represent user interest in the resources, as shown in section 2.4.1 of Chapter Two?

Resource Features  Are there any feature vectors of interval or nominal data associated with or representing resources within the software system, as defined in section 2.4.3 of Chapter Two?
**Associated Natural Language**  Do resources have any raw text or natural language associated with them, as defined in section 2.4.5 of Chapter Two?

**Lack of Data Pertaining to User Interest**  Without any data or indicators pertaining to user feedback, implicit or explicit, as described in section 2.4.1 of Chapter Two, it is not possible to determine which resources are of interest to users. Personalization is not possible.
Figure 3.4: Flowchart C, Evaluating the viability of personalization of content via collaborative filtering based on characteristics of available data.

3.2.3 Node Labels for Chart C

**Interest in Resources** Are there nominal or interval data associated with system resources that represent user interest in the resources, as defined in section 2.4.1 of Chapter Two?

**Nominal Geolocation Identifier** Is there a nominal data identifier that represents the geolocation of users?

**Determining Informational Needs Based on Geolocation** As discussed in section 2.6.9 of Chapter Two, nominal geolocation identifiers may be used to enable personalization of content via collaborative filtering.
Lack of Resource Feature Vectors and User Interest  Reaching this point in the flowchart signals a lack of either explicit or implicit resource feedback from users, as described in section 2.4.1 of Chapter Two. In addition to this, a lack of feature vectors that could be used to represent the resource that might be used to enable some inter-resource similarity measure, as described in section 2.4.3 in Chapter Two. There is not enough data pertaining to either the resources within the software system nor the users of the system to allow for meaningful personalization.
Figure 3.5: Flowchart D, Evaluating the viability of personalization of content via domain-based filtering based on characteristics of available data.

3.2.4 Node Labels for Chart D

**Interest in Resources**  Are there nominal or interval data associated with system resources that represent user interest in the resources, as described in section 2.4.1 of Chapter Two?

**Lack of Data Pertaining to User Interest**  Without any data or indicators pertaining to user feedback, implicit or explicit, as described in section 2.4.1 of Chapter Two, it is not possible to determine which resources are of interest to users. Personalization is not possible.
Figure 3.6: Flowchart E, determining if domain-based filtering is possible using natural language associated with resources.

3.2.5 Node Labels for Chart E

Vector Extraction Techniques Can **TF-IDF** or one of the methods described in section 2.5.3 of Chapter Two be used to create vector space model representations of the natural language contained within system resources?
Lack of Resource Feature Vectors and User Interest  Reaching this point in the flowchart signals a lack of either explicit or implicit resource feedback from users, as described in section 2.4.1 of Chapter Two. In addition to this, a lack of feature vectors that could be used to represent the resource that might be used to enable some inter-resource similarity measure, as described in section 2.4.3 in Chapter Two. There is not enough data pertaining to either the resources within the software system nor the users of the system to allow for meaningful personalization.
3.2.6 Node Labels for Chart F

Clickstream Data Researchers in the literature define clickstream data to be an ordered set of requests within a session representing a user’s path through a given website or web application, as defined in section 2.4.2 of Chapter Two.

Restrict Navigation Paths Should the navigational patterns be contiguous, or open, as they are described in section 2.4.7 of Chapter Two?
Contiguous Sequence Pattern Mining. Contiguous sequences, as defined in section 2.4.7, can be used to personalize the structure of software systems as shown in section 2.6.1 of Chapter Two.

Open Sequence Patterns Mining. Open sequence patterns, as they are defined in section 2.4.7, may be discovered using the algorithms and methods outlined section 2.5.8 of Chapter Two.

General Navigational Patterns Discovery. General navigational patterns, as they are defined in section 2.4.8, can be used to identify the most frequent patterns of accessed resources by users of a software system, as described in section 2.6.4 of Chapter Two.
3.3 Concluding Remarks

This chapter has presented system design and system architecture constraints, as well as their significance in relation to the specialized methods and implementations from the surveyed literature. The presented flowcharts provide the utility of directing users of the framework towards the approach and subsequent methods most likely to successfully provide personalization to a given software system based on the system’s design and architecture attributes, as well as the characteristics of available user data. If personalization is possible for a given system, the flowcharts may refer the user to a section in Chapter Two, where specific methods, algorithms, and other implementation details pertinent to the type of personalization deemed most appropriate are found. Chapter Four will apply a naturalistic data set to the framework, and discuss details of the types of personalization possible given the characteristics of the data set.
Chapter 4

Applying The Framework to a Naturalistic, Fabricated Dataset

This chapter aims to apply the framework presented in the previous chapter to a fabricated data set which closely resembles the data that would be gathered by a modern web application, and explore the steps the owners of such a site might take to personalize the software system from which the data was collected. Similarities the data set shares with input data used in systems from the literature will be examined. A method of enabling personalization for the system will be presented, along with detailed processes through which the owners of the data may achieve this solution.

4.1 The Data Set

For ease of reference, we will postulate that the data belongs to a fictional software consultancy and data analytics firm named Newman Analytics. Newman Analytics’ data set consists of server logs which contain usage data of users visiting their website over the course of several weeks, enough to collect a
large amount of information pertaining to system usage and user interactions with the site. Similar data could be realistically expected to be available for most modern websites of web applications. A sample data log – a single row within the server logs – is presented in Figure 4.1.

The framework presented in the previous chapter will be used to determine which approach to personalization is most appropriate for Newman Analytics’ website based on the characteristics of the data set, and determine which methods from the literature are most suitable for accomplishing this task.

```
{
    "UID": "2a30ef51ae762b0c4a9",
    "SessionID": "3ef04ba19",
    "Browser": "google-chrome",
    "Browser type": "google-chrome v48.2",
    "Connection": "lan/wifi",
    "Geosegmentation": "Canada/ON",
    "Operating System Family": "microsoft windows",
    "Operating System Version": "windows 10",
    "Entry Page": "home.html",
    "URL": "http://www.newmananalytics.com/account.html",
}
```

Figure 4.1: A single row within Newman’s server logs displayed in Javascript Object Notation format.

### 4.2 Analysis of the Newman Analytics Data Set

Let $D$ represent the data set.

$$D = \{d_1, d_2...d_n\}$$
where \( n \) is the number of recorded requests to the site, and \( d_i \) is a single server log representing a single HTTP request. The structure and contents of a single request are presented in Figure 4.1.

The data set contains several important pieces of information that will be vital in selecting the appropriate approach to personalization, along with suitable methods, techniques, and algorithms to enable effective personalization of the site. With only the knowledge of the examples of software systems affording personalized experiences to their users that are presented in the previous chapters, it can be deduced that the fields most crucial to enabling personalization of the site are:

1. UID
2. SessionID
3. GeoSegmentation
4. URL

These data fields represent the information that will be most valuable in determining which approach and methods from the literature are most appropriate for the personalization of this software system.

### 4.3 How Does The Newman Analytics Data Set Resemble Data in the Literature Review?

At first glance, it is clear that the data set most similarly resembles the input data used for the techniques outlined in section 2.6.1 of Chapter Two: those used to personalize structure via collaborative filtering. Both the input data used in these surveyed methods and the data set belonging to Newman Analytics are sets of sessions containing information pertaining to the number of
requests per session, the pages accessed during a session, and more. However, it is not immediately clear from the Newman Analytics data what type of personalization may be possible; it is also unclear which methods and techniques might be used to achieve such personalization. For such insights, we turn to the framework.

### 4.4 Applying the Framework - An Example

We will step through the framework – starting at the Global Entrypoint in Figure 4.2 – and explicitly state the control flow that led to the selection of a method used to implement personalization for this system. The process of using the framework will be discussed from the perspective of a Newman Analytics software architect tasked with selecting a method with which to personalize the site. The path taken at each step of the flowcharts by the owners of Newman Analytics are shaded and set in bold, and the labels for each of the figures explains the rationale behind each decision.
Figure 4.2: Because user interactions of the Newman Analytics site are navigational in nature (clicking from one resource to the next via hyperlinks), the system design constraint pertaining to navigation of structure is satisfied. The Newman Analytics architect consults section 2.4.1 of Chapter Two to determine how to gauge user interest, and determines interactions such as viewing a webpage to be indicators of interest in a resource in this system. Control flows to Chart A.
Figure 4.3: The architect tasked with personalizing the Newman Analytics site consults section 2.4.2 and learns that the data set meets the requirements for session level data. The framework informs the architect that personalization of structure via collaborative filtering is possible, and control flows to Chart F.
Figure 4.4: The architect once again consults section 2.4.2 in Chapter Two and determines that the Newman Analytics data set is not click-stream data because there are no timestamps associated with each individual request within the session-level data.

It is clear that applying the Newman Analytics data set to the framework leads to a process used to provide personalization – General Navigational Patterns Discovery. The node label for General Navigational Patterns Discovery points the architect to section 2.6.4 of Chapter Two, which details the *Clustering with Ensuing Classification* implementation method. The selected approach is chosen due to the fact that the usage data contained within the Newman Analytics data set consists of session level data, but cannot be classified as click-stream data because the HTTP requests users make do not have timestamps associated with them. Further explanations of how the hypothetical owners of the data arrived at a method to provide personalization can be found in
the labels for figures 4.3 and 4.4. The following section will provide a detailed explanation of how the Clustering with Ensuing Classification method can be applied to the Newman Analytics data set in order to personalize the site.

4.5 Personalizing the Structure of the Newman Analytics Site via Collaborative Filtering

This implementation personalizes the structure of the Newman Analytics site by providing adaptive navigation support via the clustering of general navigational patterns, a popular approach to providing scalable personalization in the surveyed literature, as outlined in section 2.6.4 of Chapter Two. Input data required for this implementation:

1. A set of sessions, where sessions are unordered sets of pageviews

The clustering with ensuing classification process:

1. **Offline Component**
   a) Process unformatted server logs
   b) Create clusters from the processed server logs
   c) Create aggregate user models from the clusters of sessions

2. **Online Component**
   a) Construct user model as the active user browses the Newman Analytics site
   b) Find the most similar aggregate user model
   c) Provide adaptive navigation support

Input and output for each of these stages are presented in Figure 4.5.
Figure 4.5: Steps used to provide adaptive navigation support using aggregate user models, and input/output for each stage. The top row represents the offline component, while the bottom row represents the online component.

1a) **Process Unformatted Server Logs** For each session within the data set, remove any non-essential information from the unformatted Newman Analytics logs, leaving only information pertaining to which resources the user accessed during their session. Information such as IP address, entry page, and browser type can be discarded. Each session will be represented as a vector of binary data points, with a cardinality equal to the total number of pages on the Newman Analytics site. Each dimension in these vectors is associated with a page on the site. In the vector created for each session, the value at each
dimension is 1 if the user viewed the page during their session, 0 if they did not. Let this set of vectors be $S$.

1b) Create Clusters From the Processed Server Logs $S$ will be used as input to a clustering algorithm, such as $k$-means. $k$-means is a popular clustering algorithm for the task of creating clusters which are subsequently used to create aggregate user models because the number of clusters and therefore, the scalability of the online component are controlled. Cosine similarity or an equivalent similarity measure from section 2.5.2 in Chapter Two may be used to determine how similar vectors are to each other. This offline component will generate $k$ clusters, which aim to group sessions together based on their access patterns on the Newman Analytics site. Session clustering will enable the discovery of pages on the Newman Analytics site that are commonly accessed together during a single session.

1c) Create Aggregate User Models For each cluster, the mean vector is calculated from all vectors within that cluster – the cluster centroid. This mean vector is used to represent an aggregate user model, representing an average, or typical user within each cluster. Again, within the aggregate user model, dimensions within the vector are associated with pages on the Newman Analytics site, and the values at those dimensions pertain to the mean interest level a user in that cluster has in the associated page on the site. These aggregate user models will reveal general navigational patterns within the Newman Analytics data set, as they are described in section 2.4.8 of Chapter Two.

With this approach, the online component will only need to examine $k$ aggregate user models to find which one the active user most closely resembles,
\[ AUM_1 = \{0.82, 0.77, 0.22, 0.91, 0.31, n_1\} \]

Figure 4.6: A single aggregate user model \( AUM_1 \) is the mean vector within a given cluster. Users in this cluster typically view the page associated with the first, second, and fourth dimensions of this vector, but not the third, nor the fifth (higher values indicate a larger proportion of users within the cluster interacting the the resource associated with the vector dimension).

and thus infer what their informational needs are and what pages are most likely to be of interest to them.

2a) Construct a User Model For The Active User The online component will run in real time, and build a user model as an active user browses the Newman Analytics site. The user model is a vector of boolean data points, where the value at a given dimension represents whether or not the active user visited the page during their session. Pages are tracked in a vector in the user model, which has the same cardinality as the vectors processed by the offline component.

2b) Find The Most Similar Aggregate User Model Since the aggregate user models generated in the offline component and the active user’s user model are both vectors of size \( n \), where \( n \) is the number of pages on the Newman Analytics site, determining which aggregate user model most closely resembles the active user’s user model is trivial. Using the same similarity measure used by the offline component in conjunction with the \( k \)-Nearest Neighbours algorithm – with \( k = 1 \) – will yield the most similar aggregate user model. Once the most similar aggregate user model \( AGM_x \) is found, the pages in \( AGM_x \) associated with the highest values that have not yet been viewed by the active user are added to the set of candidate resources for recommendation, \( R \).
2c) **Provide Adaptive Navigation Support**  Recommend pages within the set of candidate resources, \( R \), using any of the adaptive navigation support techniques outlined in section 2.2.2 in Chapter Two. Since we assume that similar users share similar informational needs, pages that are contained within the aggregate user model most similar to the active user are the most likely to be useful in satisfying the active user’s informational needs.

### 4.6 Concluding Remarks

This chapter has presented the application of a fictional but naturalistic data set to the framework presented in Chapter Three of this thesis, which itself is based on the research of the surveyed literature. The presented approach, which was chosen by applying the framework to the Newman Analytics data set, can be used to personalize the fictional Newman Analytics site. Consequently, negative implications of the information overload problem for users of the site are reduced. Section A.2 of the appendix offers alternate implementations of personalization of structure via collaborative filtering that could be feasible if the Newman Analytics dataset contained click-stream data. This chapter confirms that the framework can indeed be used to determine a suitable approach to personalization based on the characteristics of available data.
Chapter 5

Final Remarks and Future Work

This thesis has demonstrated the value of personalization in countering the adverse effects of the information overload problem. The rise of the internet as a tool for the distribution of information has led to an overabundance of materials, knowledge, and entertainment with which users of contemporary software systems are regularly inundated. The role personalization plays in combating these bombardments of irrelevant information was detailed in Chapter One of this thesis, along with an overview of how this task can be accomplished. Chapter One also established the potential utility of a framework which could enable owners of software systems to determine the most effective methods to enable personalization for a given system from the plethora of techniques and implementations in the literature, based on the characteristics of available data.

Chapter Two of this thesis presented a comprehensive review of the current trends pertaining to the personalization of existing systems from the literature provide, along with the methods, techniques, and algorithms these systems utilize. The characteristics of the data these systems use as input were analyzed.
To simplify the identification of these trends within the literature, software systems were analyzed from the perspective of two themes: goal of personalization and method of personalization. With personalization of structure and personalization of content being the two dimensions through which goal of personalization may be analyzed, and collaborative filtering and domain-based filtering representing the two dimensions through which we may observe method of personalization. Representing these themes as a matrix – as is done in Figure 2.1 in Chapter Two – yields four different approaches to personalization. These are, once again, personalization of structure via collaborative filtering, personalization of structure via domain-based filtering, personalization of content via collaborative filtering, and finally personalization of content via domain-based filtering. Analyzing software systems from these dimensions allows for simpler categorization of these systems and subsequent discovery of trends in methods, techniques, algorithms and data requirements for personalization.

Chapter Three presented the system design and system architecture constraints through which individuals seeking to provide personalization to a given software system might evaluate which approach to personalization is likely to be most successful for a given system. Chapter Three also presented the flowcharts which comprise the framework. Through a series of questions regarding the characteristics of available data, the most effective approach to enabling personalization is discovered. Furthermore, the most appropriate methods and techniques for a given system are determined.

Chapter Four applied this framework to a naturalistic data set belonging to Newman Analytics, which contains data similar to what contemporary websites collect about user interactions with their technology, demonstrating the utility and practicality of the framework. The manner in which a user might
personalize a software system was described in the context of the fabricated Newman Analytics data set and the site from which the data was collected, describing the actions and operations required to enable personalization on the site.

As a whole, this thesis presents relevant information from the literature that pertains to the methods and techniques used to personalize software systems, and the myriad of data requirements these approaches necessitate. The categorization of these techniques from the literature indicate that trends regarding the relationship between the type of personalization a given software system provides and the data these systems use to enable such personalization certainly exist. The presentation of a framework that formalizes these established trends can be of significant assistance to those seeking to enable personalization for their software system. This is especially true for those who do not have extensive knowledge of state of the art methods of personalization, as it can greatly facilitate the search for an appropriate method or technique that can be used to adequately and effectively provide personalization.

The goal of creating such a framework is successfully accomplished. The practicality and usefulness of the framework is also demonstrated by the application of the fabricated dataset belonging to consulting firm Newman Analytics to the framework. However, only the application of real datasets – belonging to real software systems – to this framework will prove its true utility and viability as a functional tool. Doing so will undoubtedly provide useful feedback with which the questions contained in the flowcharts can improve the manner in which the framework’s user is queried about the characteristics of their software system and the data it makes available.
Future work in extending this framework could be centered on researching the lesser utilized hybrid approach to filtering, which is rooted in both collaborative and domain-based filtering. Continued research could also determine which types of software systems might be appropriate candidates for this method of personalization as well as what data requirements the method might impose. Additionally, an investigation into the personalization methods used by systems that are less academic in nature, such as those used by Amazon™, Netflix™, Pinterest™, and Spotify™ would likely yield very advanced and specialized methods and techniques of enabling personalization, and provide useful insights into the corporate world of enterprise-level personalization. Such insights could be utilized to increase the dependability of this framework, and could serve to refine and enhance the lens through which it analyzes available data.
Appendix A

Methods and Techniques

A.1 Pseudocode For Obtaining Session Level Data From
the Newman Analytics Data Set

This section presents pseudocode which describes a suggested implemen-
tation for preprocessing the Newman Analytics Data set in order to obtain a
session level data set. This session level data set is described in Chapter Four
as $S$.

In the following pseudocode, $\{\}$ represents a dictionary, and the $+$ operator
denotes the concatenation operation. As outlined in Chapter Four, $D$ is the
Newman Analytics data set.

\begin{verbatim}
let $S \leftarrow \{\}$
for each row in D do
    let id $\leftarrow row.visitorID$
    let visitNumber $= row.visitNumber$
    let key $\leftarrow visitorId + \_ + visitNumber$
    if $S[key] \neq nil$ then
\end{verbatim}

101
A.2 Alternate Personalization Methods For Newman Analytics

This section of the appendix will present other methods of personalization that could be used by Newman Analytics if the characteristics of their data set were different from those presented in Chapter Four. For each of the following subsections, the decision taken in Chart F by the Newman Analytics software architect will be presented.
Figure A.1: In this scenario, we hypothesize that the sessions contained within the Newman Analytics dataset contain timestamps of each HTTP request. Additionally, the software architect chooses to restrict navigational paths in order to make sure users are only recommended following specific paths, as per the description in section 2.6.5 Chapter Two.

A.2.1 Adaptive Navigation Support Using Sequential Patterns

This implementation personalizes the structure of the Newman Analytics site by discovering frequent navigational patterns users established by users, and providing adaptive navigation support to new users who display the patterns of past users. Input data required for this implementation:
• A set of sessions, which are in turn ordered sets of pageviews, with each pageview containing a nominal URI identifier

1. **Offline Component**
   a) Sessionize and preprocess raw server logs
   b) Discover most common paths using a frequent itemset mining algorithm

2. **Online Component**
   a) Construct user model as the active user browses the Newman Analytics site
   b) Provide adaptive navigation support by finding most similar paths

Input and output for each of these stages is presented below, in Figure 4.5.

Figure A.2: Steps used to provide adaptive navigation support using Sequential Patterns, and input/output for each stage. The top row represents the offline component, while the bottom row represents the online component.

1a) **Sessionize and Preprocess Raw Server Logs**  By preprocessing the server logs comprising $D$ using the pseudocode found in section A.7 of the
appendix, a set of sessions, (user visits to the site) are obtained. A single session is represented as a set in the following format:

\[
s = \{p_1, p_2, \ldots, p_n\}
\]

Where \( n \) is the number of requests the user made to the Newman Analytics site during their session. The values within this set are records containing a nominal URI by which the page the user requested may be identified, as well as a timestamp pertaining to when the request was made. Because this implementation seeks to explore navigational paths, ordering of requests to the site by time of request is required. Once this ordering is complete, we have:

\[
S = \{
\{p_1, p_2, \ldots, p_n\}_1,
\{p_1, p_2, \ldots, p_n\}_2,
\{p_1, p_2, \ldots, p_n\}_3,
\{p_1, p_2, \ldots, p_n\}_m
\}
\]

Where \( S \) is the set of all sessions, and is composed of ordered sets, and \( m \) is the number of sessions represented as ordered sets.

1b) Discover Most Common Paths  Frequent itemset mining algorithms such as PrefixSpan, CloSpan, or AprioriAll may be used to determine the most common navigational paths within the set of sessions, \( S \), as outlined in section 2.5. of Chapter Two. These algorithms may be configured to mine either open or closed sequential patterns, depending on what the personalization goal is. Open patterns do not require that patterns are an exact match, and are better suited for recommendation while while closed patterns require both patterns to
be identical to match, and are more appropriate for predicting the next page a user will visit. Refer to section A.6 of the appendix for an in depth explanation of the difference between open and closed sequential patterns. As discussed in Chapter Two, these algorithms will create a set of sequential paths that occur within $S$ at or above a specified frequency, known as a support threshold. The result of this stage is the set of all frequent sequential patterns, $F_s$.

2a) Construct a User Model for the Active User The previous steps represent the offline component of this implementation. The online component consists of tracking the pages the active user visits as they browse the Newman Analytics site.

As the active user browses the site, a nominal URI representing the page is added to an ordered set containing the pages the user has browsed.

2b) Provide Adaptive Navigation Support With each new page request, the user model $U$ belonging to the active user is sent to the server. As displayed in section 2.5 of Chapter Two, the set of frequent sequential patterns $F_s$ created by the offline component is searched for frequent sequential patterns that are of cardinality $|U| + 1$ that contain $U$ as their prefix. To ensure this process is as efficient as possible, the prefix tree described in Chapter Two of this thesis should be used to store frequent sequential patterns. All matching patterns of size $|U| + 1$ are sorted by the confidence in the association rule mined from $F_s$ in which the antecedent is the active user’s current session, and the consequent is the matching pattern of size $|U| + 1$. The suffix of these patterns (the resource that has not yet been viewed by the active user) are added to a list of candidate resource recommendations, $R$. From $R$, the top $n$ links that can be reasonably fit to the UI are recommended via link generation or an equivalent technique, and provide adaptive navigation support to the users of the site.
A.2.2 Adaptive Navigation Support via Association Rule Mining for Revealing Common Itemsets

Chart F

Determining the most suitable methods and implementation details to personalize structure via collaborative filtering.

Figure A.3: In this scenario, we hypothesize that the sessions contained within the Newman Analytics dataset contain timestamps of each HTTP request. The software architect chooses not to restrict navigational paths in order to make sure users are recommended any pages to the active user that past users have found interesting, regardless of chronological page view ordering, as per section 2.6.5 in Chapter Two.

This implementation personalizes the structure of the Newman Analytics site by determining which resources users of the site frequently access together during individual sessions, and recommends resources to new users to the site.
based on how similar the resources they access are to those accessed by past users of the site. Input data required for this implementation:

1. A set of sessions, where sessions are unordered sets of pageviews

1. **Offline Component**
   a) Preprocess raw server logs
   b) Discover Most Common Patterns

2. **Online Component**
   a) Construct user model as the active user browses the Newman Analytics site
   b) Provide adaptive navigation support

Input and output for each of these stages is presented below, in Figure 4.7.

![Figure 4.7: Steps used to provide adaptive navigation support using Common Itemsets discovered via Association Rules Mining, and input/output for each stage. The top row represents the offline component, while the bottom row represents the online component.](image-url)

108
1a) Preprocess Raw Server Logs  Create a set of user sessions $S$ from the raw server logs $D$, as described in section A.7 of the appendix. Unlike the implementation provided in section 4.5.1 of this chapter, *Adaptive Navigation Support Using Sequential Patterns*, this approach is not concerned with sequential patterns but instead with resources frequently accessed together during a single session, regardless of the order in which the pages were accessed.

1b) Discover Most Common Patterns  As is the case with the mining of common paths described in section 2.6 in Chapter Two, the Apriori algorithm and variations of the FP-Growth algorithm can be used to find common itemsets within the processed Newman Analytics session logs, $S$. In this context, itemsets are sessions, with items or elements of the set being pages that were accessed during the session. Pages are to be identified by their formatted nominal Uniform Resource Identifier, or some internal reference to the URI.

As with the implementations from the literature described in Chapter Two of this thesis, mining the dataset using these or equivalent techniques will uncover the sets of pages that are most frequently accessed together, as long as the frequency with which they are accessed together during a given session meets or exceeds a given threshold. This threshold is known as the support of an itemset. Let the result of this stage be $F$, the set of all frequent patterns which have some given level of support, or frequency of occurrence within $S$. Note that this set of frequent patterns $F$ is not the same as the set of frequent sequential patterns $F_s$ discussed in section A.6 of the appendix.

Frequent itemsets from $F$ are sorted lexicographically and inserted into a directed acyclic graph, in which the depth or nodes within the graph is equivalent to the cardinality of the frequent itemset being stored, as displayed in Figure 2.2. Nodes have edges to nodes containing itemsets which are supersets
of themselves and contain one additional element within their itemset. This is
done in order to ease the matching of an active user’s session with elements
within $F$ by the online component.

2a) Construct a User Model for the Active User  As the active user
browses the site, references to the pages they have viewed on the Newman
Analytics site during the session are stored within their user model, sorted
lexicographically, and not by order of access. Because this approach is con-
cerned with frequent patterns of page accesses and not frequent navigational
paths, pages are not to be stored in chronological order of access, but rather in
lexicographical order. The information stored in the active user’s user model
is thus a lexicographically ordered set of references to the pages the user has
browsed on the Newman Analytics site. References to pages should be in the
same format as those used by the association rules mining performed by the
offline component.

2b) Provide Adaptive Navigation Support  The active user’s user model
is sent to the server with each new page request. The directed acyclic graph
containing lexicographically sorted frequent itemsets is searched for an itemset
$M$ matching the active user’s user model. If such a match is found, the nodes
to which $M$ has edges (if any) contain frequent itemsets that contain the active
user’s session, along with one more resource, as they are one level deeper in
the graph. For each frequent itemset $f$ to which $M$ has an edge, the confidence
within $F$ that the association rule where $M$ is the antecedent and $f \cup M$ is
the consequent is determined. If the confidence in the rule meets or exceeds
a chosen threshold (unfortunately, these authors do not specify the threshold
used), the antecedent is added to the list of candidate resources, $R$. Pages
in the list of candidate resource recommendations are sent back to the client
and used to provide adaptive navigation support using link generation or an equivalent technique. The graph constructed during the offline component allows for frequent itemsets to be discovered with great efficiency, enabling recommendations to be provided in real time.

References


computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing (cit/iucc/dasc/picom), 2015 ieee international conference on (pp. 395–402).


Baraglia, R., & Silvestri, F. (2007). Dynamic personalization of web sites


Beel, e. a., Joeran. (2013a).

Beel, e. a., Joeran. (2013b).


Berendt, B., Mobasher, B., Nakagawa, M., & Spiliopoulou, M. (2002). The impact of site structure and user environment on session reconstruction
in web usage analysis. In *International workshop on mining web data for
discovering usage patterns and profiles* (pp. 159–179).

Bianco, A., Mardente, G., Mellia, M., Munafò, M., & Muscariello, L.


Bobadilla, J., Ortega, F., Hernando, A., & Bernal, J. (2012). A collabora-
tive filtering approach to mitigate the new user cold start problem.

*Knowledge-Based Systems, 26*, 225–238.

Bozdag, V., & Timmermans, J. (2001). Values in the filter bubble ethics
of personalization algorithms in cloud computing. In *1st international
workshop on values in design–building bridges between re, hci and ethics,
lisbon, portugal, 6 september 2011*.

modeling and user-adapted interaction*, 6(2-3), 87–129.

290.

Brusilovsky, P., & Maybury, M. T. (2002). From adaptive hypermedia to the

Chaudhary, K., & Gupta, S. K. (2013). Web usage mining tools & techniques:
4(6), 1762.

Choi, K., Yoo, D., Kim, G., & Suh, Y. (2012). A hybrid online-product
recommendation system: Combining implicit rating-based collaborative
filtering and sequential pattern analysis. *Electronic Commerce Research*


workshop on web information and data management (pp. 88–95).


