Matching Recommendation Technologies and Domains

Robin Burke and Maryam Ramezani

Abstract Recommender systems form an extremely diverse body of technologies and approaches. The chapter aims to assist researchers and developers identify the recommendation technology that are most likely to be applicable to different domains of recommendation. Unlike other taxonomies of recommender systems, our approach is centered on the question of knowledge: what knowledge does a recommender system need in order to function, and where does that knowledge come from? Different recommendation domains (books vs condominiums, for example) provide different opportunities for the gathering and application of knowledge. These considerations give rise to a mapping between domain characteristics and recommendation technologies.

1 Introduction

Unlike some other types of software systems, recommender systems are not defined by a particular kind of computation, like for example, a statistical computation package, or by the storage and use of a particular kind of data, as in a geographical information system. A recommender system is defined by a particular kind of semantics of interaction with the user: "any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options" [1]. This expansive definition makes the scope of recommender systems research quite broad, but it fails to give much guidance to the implementer. A crucial question is therefore how

Robin Burke
Center for Web Intelligence, School of Computer Science, Telecommunication and Information Systems, DePaul University, Chicago, Illinois, USA e-mail: rburke@cs.depaul.edu

Maryam Ramezani
Center for Web Intelligence, School of Computer Science, Telecommunication and Information Systems, DePaul University, Chicago, Illinois, USA e-mail: mramezani@depaul.edu

recommendation techniques can be matched to recommendation problems. That is the question that this chapter tries to address.

2 Related Work

There are several taxonomies for recommender systems. Burke [1] distinguishes between five different recommendation techniques: collaborative, content-based, utility-based, demographic, and knowledge-based. The article discusses the advantages and disadvantages of each technique and proposes hybrid recommender systems to gain better performance with fewer of the drawbacks of any technique in isolation. An early work in recommender systems [2], which focuses on collaborative recommenders, identifies 5 dimensions to place the systems in a technical design space. The dimensions characterize properties of the users' interactions with the recommender and the aggregation methods of users' evaluations (ratings). Konstan and Schafer [3] present a taxonomy of collaborative e-commerce recommender applications that separates their attributes into three categories: the functional I/O, the recommendation method, and other design issues such as degree of personalization and delivery methods. An eight-dimensional taxonomy of recommender systems is presented in [4] using two main criteria: user profile generation and maintenance, and user profile exploitation techniques. In this taxonomy, common patterns in recommender systems are extracted by an analysis of the systems in the same domain. A more recent survey on recommender systems [5] classifies recommendation methods (omitting knowledge-based) into three main categories: content-based, collaborative, and hybrid recommendation approaches and classifies recommenders in each category into either heuristic-based or model-based.

This work is distinguished from previous categorizations in that it is not aimed at classifying existing recommender systems along particular dimensions of interest as in the surveys above, but rather as an AI-centric approach, focused on the knowledge sources required for recommendation and the constraints related to them. The chapter discusses the applicability of different recommendation techniques to different types of problems and aims to guide decision making in choosing among these techniques. As such, it might be considered to serve as a sort of recommender for recommender system implementers.

3 Knowledge Sources

A fundamental choice for an implementer of an AI system is the source and type of knowledge that the system will employ. In the case of recommender systems, there are two primary entities about which we might have knowledge: because recommendation is personalized, a recommender must have knowledge of its users; the recommender may also have knowledge about the features of the items that it

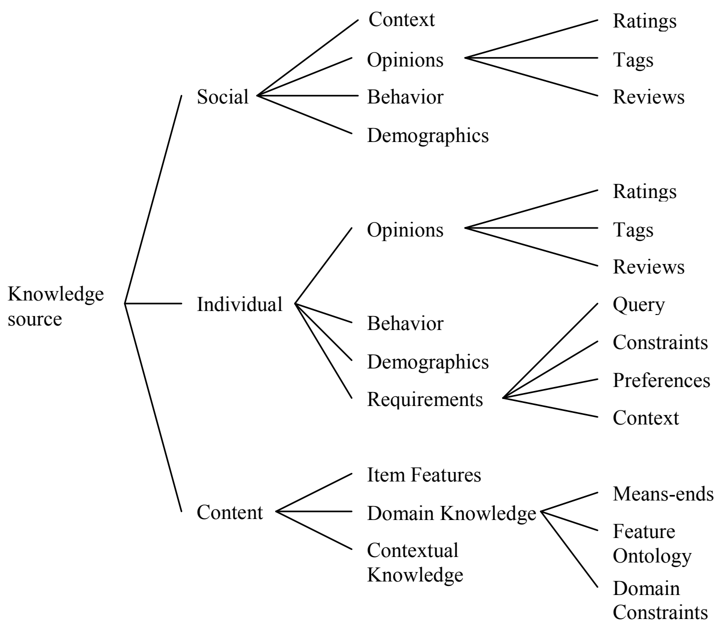

Fig. 1 Taxonomy of knowledge sources in recommendation

is recommending. It is considered one of the chief benefits of collaborative recommendation that domain knowledge or item features are not required, but all other recommendation techniques require them.

For an individual instance of recommendation, we are presented with a particular target user and seek to make personalized recommendations for him or her. In this situation, we can divide the knowledge of users into what we know about the target user, and what knowledge we have of the user community at large. There are therefore three broad categories of knowledge that may come into play in recommendation:

- **Social:** Knowledge about the larger community of users other than the target user.
- **Individual:** Knowledge about the target user.
- **Content:** Knowledge about the items being recommended and, more generally, about their uses.

Felfernig and Burke present a taxonomy of recommendation knowledge in [6]. Figure 1 shows this taxonomy further expanding each category into subtypes of knowledge, all of which have been used in some existing recommender systems. These subtypes are explained below.

Social knowledge is the total sum of all of the user profiles stored in a system. Collaborative recommendation is intensive in its application of social knowledge, usually with profiles of the simplest type: user opinions such as the liked-disliked scales used in MovieLens and other well-known collaborative recommenders, or interaction histories as seen in collaborative web personalization. Other types of knowledge can come into play, however. In the I-SPY collaborative web search application, user's queries (requirements) are recorded as well as their preferences (link selections) relative to those queries [7]. Demographic information about the user base is also employed by some recommender systems.

Individual knowledge is what drives a given interaction with the recommender system. It may be relatively implicit, in the sense of a user's profile being recalled to memory and used to initiate processing of recommendations, or it may be explicit in that the user specifies his or her interest before or during the recommendation process. In addition to the behavioral, opinion and demographic types of knowledge described relative to the social category, individual requirements are much more commonly found. For example, in the class of knowledge-based recommender systems known as critiquing systems, a user examines recommended items and responds with multi-dimensional critiques that act as constraints and/or preferences on the next round of retrieval [8]. Such systems often begin their interaction with a query that the user formulates.

Content knowledge has a variety of forms. In its simplest incarnation, the system might only have knowledge about the features of items that it is recommending, enabling it to learn what features a user seems to prefer. Domain knowledge refers to more complex notions of content knowledge such as means-ends knowledge, that is what means/features are appropriate for which goals/ends that the user might have in mind. A feature ontology relates features to each other so that similarity and difference between items can be more adequately assessed. Domain constraints may be necessary to prevent a system from recommending an item that is inconsistent with what the domain permits.

Context is an important factor in many application domains. What makes for an appropriate recommendation will often be the function of the context – a search for a restaurant to conclude a high-powered business transaction will be different from a search a search aimed at finding a place to celebrate a four-year-old's birthday even if the searcher's profile, and her query, "Italian" are the same. The use of context in different types of recommender algorithms including collaborative recommendation is an area of active research [9, 10, 11, 12, 13, 14]. However, there is no consensus on how best to profit from it or even how to define the term. Contextual issues in recommendation will receive limited treatment in this chapter.

3.1 Recommendation types

Recommendation types are explained in detail elsewhere in this volume. However, in conjunction with a discussion of knowledge sources, it is worth considering how different recommendation types operate.

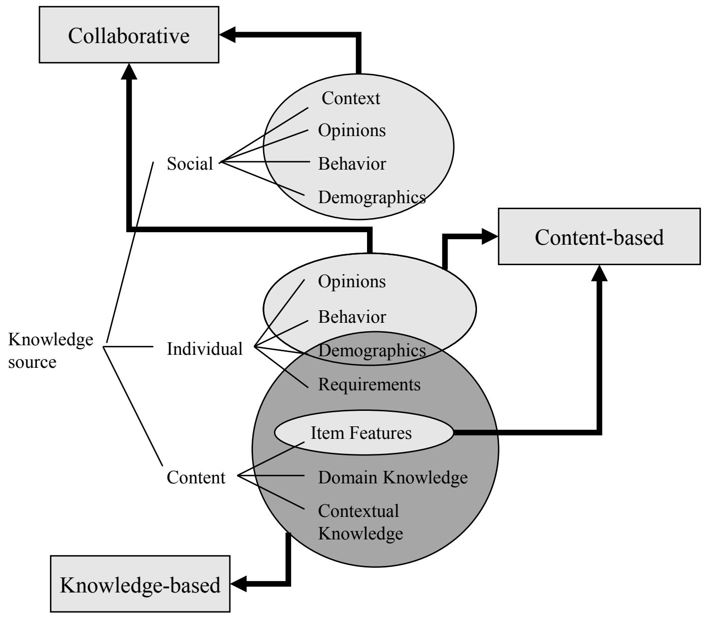

Fig. 2 Knowledge sources and recommendation types

- Collaborative recommendation matches an individual knowledge source with a social knowledge source of the same type and extrapolates the target user's preferences from his or her peers. Usually, in collaborative recommendation, individual requirements are not used, or applied very simply as filters.
- Content-based recommendation on the other hand is individually-focused, using item features and user opinions to learn a classifier that can predict user preferences on new items.
- Knowledge-based recommendation is more of a catch-all category in which the recommender applies any kind of domain knowledge more substantive than item features.

Figure 2 shows the connection between knowledge sources and the recommendation types.

From a knowledge source perspective, hybrid recommendation, which is discussed elsewhere in this volume, is really a matter of combining knowledge sources that have not traditionally been put together in the three types discussed above. Often, a hybrid is created by adapting an algorithm for one recommendation type to accept a knowledge source more typically associated with another type.

Of course, a knowledge source does not a recommender system make. An AI system also needs algorithms. It is difficult to generalize since new recommendation algorithms are put forward with great regularity, but in general, collaborative systems use multi-class classification algorithms for extremely sparse and high-dimensional spaces; content-based systems use binary learning algorithms for lower-dimensional spaces; and knowledge-based recommenders use inference schemes of various types.

Still, an algorithm can only function with the right knowledge sources, and it is on this topic that this chapter will concentrate. Considerations about the domain of application and the style of interaction with the user lead us to conclusions about the availability and characteristics of different knowledge sources. These considerations in turn can be used to guide the selection of feasible recommendation algorithms. We turn next to the characteristics of domains.

4 Domain

A domain of recommendation is the set of items that the recommender will operate over, but may also include the set of aims or purposes that the recommender is intended to support. A specialized recommender, for example, a news recommender that identifies stories for the attention of government intelligence analysts, may have different implementation considerations than a generalized news recommender such as Google News. In turn the characteristics of the domain affect the availability and utility of different knowledge sources. In the online news case, there are a huge number of news sources and articles such that no user will never have time to experience or rate more than a small fraction of them. In addition, the news itself is undergoing constant change. So, we can characterize the ”Social / Opinions / Ratings ” knowledge source as one of great sparsity and great dynamism.

Another aspect of the domain has to do with the larger application in which the recommender is embedded. If the recommender is a part of a larger system like an e-commerce site, it may be necessary for the recommender to impose as little as possible on the normal user interaction with the application, which means the system has to use implicit user inputs. On the other hand, if the recommender is the primary application that users are accessing, it can gather data explicitly from users.

We have identified six important characteristics of the domain that an implementer should consider: heterogeneity, risk, churn, interaction style, preference stability, and scrutability.

4.1 Heterogeneity

A heterogeneous item space encompasses many items with different characteristics and most importantly, different goals they can satisfy. For example, an e-commerce recommender system as found at Amazon.com has a large number of heterogeneous items that can be recommended. Even within a single category like books, such disparate categories as home repair, romance novels, cooking, and children’s fantasy all coexist in the database.

A homogeneous recommendation space means that content knowledge relative to the domain will be easier to acquire and maintain. Consider a site that only recommends digital cameras versus one that has all kinds of electronics. The camera-only site would be able to invest in content knowledge specific to photography, whereas the general site would have a much more challenging task trying to do knowledge engineering for all of consumer electronics. Even a simple catalog of item features becomes difficult to design effectively if the items differ wildly from each other.

4.2 Risk

Recommendation domains can be distinguished by the degree of risk that a user incurs in accepting a recommendation. A 99 ¢ music track is low risk; a $1.5 million condominium or a medical diagnosis could be very high risk. Risk determines the user's tolerance for false positives among the recommendations. In some domains, false negatives may also be important – if there is a cost or risk associated with not considering some options.

Another way to think of a high-risk domain is that there are likely to be some important constraints on a valid solution that the recommender system must obey. For example, a condominium buyer is likely to have some very strong constraints about location, price and amenities. As mentioned above, the tolerance for false positives is going to be low for high-risk items.

4.3 Churn

Recommender systems are used in domains with long-lived items like books, but they are also used in domains where the value or relevance of an item has a very short time span, such as news stories. A high churn domain is one in which items come and go rapidly.

In such a domain, a recommender system faces a continual stream of new items to be integrated into its knowledge sources. This greatly increases the sparsity of any kind of opinion data, as new items will necessarily have been seen by very few users. Items that have been around for some time may accumulate ratings, but by the time they do, they may no longer be relevant.

4.4 Interaction Style

In systems in which the user makes no special effort to interact with the recommender system, the system extracts the implicit expressed preferences from user behaviour. For example, when visiting a web site, a user leaves behind behavioral traces in web server logs that can be used to make recommendations. Implicit inputs may include the specific items that the user is currently viewing, user transaction history, elapsed time, and shopping cart / purchasing behavior. Explicit inputs require that the user make the effort to formulate an opinion or a query or to add personal data to the system. There must be some perceived benefit for doing so that justifies the effort.

Implicit inputs are naturally noisy because they are inferred from user behavior. This type of interaction may be best suited for gathering simple rating knowledge, although some researchers have explored the extraction of preferences and even domain knowledge from implicit data [15, 16]. Explicit inputs may be more sparse if the burden of generating them causes users to do so relatively rarely.

4.5 Preference stability

User preferences can also have varying degrees of duration. For example, a person buying a digital camera would typically switch preferences after purchase, since they would be no longer interested once the purchase was complete, while a person interested in comedy movies may wish to continue getting comedy recommendations for a long period of time. Also, preferences for some items may increase and wane naturally, for example, when one's favorite basketball team is in a big tournament, stories about it become highly preferred, but if they are knocked out or when the tournament is over, the user's preferences will change.

Stable preferences mean that opinion data collected in the past is still likely to be valid today. This makes it easier to maintain high-quality knowledge sources that use this data. Unstable preferences mean that any data collected in the past may have to be discounted or discarded. This increases the importance of gathering the user's specific requirements of the moment. If the user generates a large amount of implicit data in a single session, then it may be possible to use individual sessions as profile data.

The problem of preference instability can be ameliorated by collecting more data. If a user generates enough opinion data during a single session to adequately represent his or her current preferences, then there is no need to extrapolate from historical data and the issue of preference stability does not arise. This situation is found in web personalization applications. Users generate a large number of clicks while browsing a web site, enough implicit data to allow for recommendation using only the data from a single session.

4.6 Scrutability

Certain applications (for example, high-risk ones) may require that the system be able to explain its recommendations, to answer questions like "why was this item recommended?" Such explanations enhance user confidence that a recommendation is appropriate [17] and increase the liklihood of recommendations being accepted.

Explaining recommendations is most straightforward when content knowledge sources are employed [18, 19]. Explaining a recommendation based on social knowledge has proved more challenging. See Herlocker et al. [20] for an evaluation of some alternatives.

5 Knowledge Sources

As we have shown above, the choice of domain and the characteristics of the application place certain constraints on the kinds of knowledge sources that a recommender system may deploy. In turn, the availability and quality of knowledge sources influences what recommendation technologies a recommender can profitably use.

5.1 Social Knowledge

Social knowledge enables the use of collaborative algorithms in which predictions about individuals are extrapolated from their peers opinions. Ratings are the most straightforward type of data used to model this knowledge. Rating knowledge is often conceptualized as a $m \times n$ matrix where m is number of users and n is the number of items and each entry corresponds to a user's rating of an item. Model-based techniques use this matrix to create a model in advance, whereas memory-based techniques use it at the time of recommendation generation to produce the prediction.

The use of other types of opinion data is an area of active research. User tags are a promising source of opinion knowledge for recommendation [21, 22, 23, 24, 25]. In this case the data will be a tripartite graph of user, item and tag. Textual data in the form of reviews are also been studied, for example in [26]. Multi-dimensional knowledge sources such as these present a challenge for existing collaborative algorithms [9].

In heterogenous domains, social knowledge should be considered as a knowledge source since it is gathered by user's input and does not need extensive knowledge engineering. However, social knowledge is not sufficiently accurate and reliable for high risk domains or for domains which need explanation.

Social knowledge will tend to be sparse for high churn domains. When items come and go quickly, the odds are reduced that any given user will have a chance to rate any given item. As with other sparsity effects, the problem of churn can be ameliorated by having a large user population. Google News, for example, can take advantage of the site's large and active user base to implement collaborative recommendation even for the high-churn news domain. [27]

Using social knowledge is appropriate for domains with implicit type of interaction since it is possible to mine the users' behavior using machine learning and statistical techniques which are the typical algorithms in collaborative filtering. In domains with unstable user preference, the social knowledge can be misleading since the historical data is unreliable.

5.2 Individual

Individual knowledge is essential requirement for a recommender system to produce personalized recommendations.

In collaborative recommendation, individual knowledge regarding the target user is matched against social knowledge drawn from the user population at large. The most straightforward version of the process is that these sources are of the same type, and all that is needed is a similarity metric by which individuals can be compared. Ratings work well for this, but researchers have also used demographic data. Krulwich [28] uses demographic groups from marketing research to suggest a range of products and services. In other systems, machine learning is used to train a classifier based on demographic data [29]. In heterogenous domains, it might be difficult to transfer user's input on certain items for recommending other items. For example, it is not certain that two users who have similar taste about movies, would also like similar music.

In the absence of social knowledge, individual knowledge especially in the form of ratings can also be combined with content knowledge in the form of item features to build a classic content-based recommender that uses supervised classification learning [30, 31, 32].

A domain that requires knowledge of the user's short-term requirements is most likely suited to some kind of knowledge-based recommendation. The query is the most fundamental form of input for requirements: the user states, in whatever form the system accepts, what it is they are looking for. Constraints and preferences allow the user to limit and to rank options. For example, a dog owner might have a strict constraint that any apartment he rents accept his pet. A parent with young children might have a preference to be close to parks and playgrounds. In high risk domains and domains which need explanation, it is usually necessary to have explicit requirements and constraints from the user. Similarly, user requirements are more likely to be needed in domains with unstable preferences since the historical data are unreliable.

In many recommender systems more than one type of user input is used. For example, [33] uses users' priorities and constraints in a CBR recommender. A survey of preference elicitation methods can be found in [34].

5.3 Content

The most basic kind of content knowledge is item features, the kind of data that would typically be available in a product database, such as numeric values like price or symbolic tokens (destination airport, for example). These features can typically be used as is in a recommender system, although implementers will often want to restrict the feature space. For example, the entire cast and crew list for a movie may be available as feature data but will contain many sparse features of little utility. Trimming this list to just the top billed actors, director, and screenwriter would probably be sufficient.

If items are represented by unstructured documents such as news stories, the implementer will need to draw from information extraction (IE) techniques to extract and select features for use in recommendation. Standard techniques include eliminating stop words, stemming to simplify the feature space. Features can be reduced further by applying more sophisticated feature selection techniques such as information gain, mutual information, cross entropy or odds ratio [35]. More structured documents such as HTML pages offer additional opportunity for feature extraction. Applications of IE techniques to extract content knowledge from semi-structured and structured documents are discussed in [36]. Content knowledge in multimedia format presents an additional challenge. Hauptmann [37] discusses techniques from multimedia information retrieval. Osmar et al. survey multimedia data mining in [38] with a number of techniques useful for recommendation.

The quality of recommendations produced by a content-based or knowledge-based recommender will be entirely dependent on the quality of the content data on which its decisions are based. Indeed, the lack of reliable item features is often cited as a motivating factor for avoiding content-based recommendation. The cost involved in creating and maintaining a database of useful item features should not be underestimated, particularly for heterogeneous domains. In a domain like digital cameras or cell phones, for example, new technical innovations arrive regularly, requiring that the schema and the individual entries for each item be updated. If there are a large number of not-entirely-independent features extracted in a variety of ways, the system may be tolerant of noisy feature data. On the other hand, applications with high risk will need to pay special attention to having clean item features. Typically, manual review of feature data or manual labeling will be required.

5.3.1 Domain Knowledge

A knowledge-based recommender will typically need to know more than just what features are associated with what items. The most basic form of domain knowledge that a recommender can employ is an ontology over the item features. Such an ontology allows the system to reason about the relationship between features at a level deeper than just raw equality or difference. For example, the restaurant recommender Entree [8] has an ontology of different types of cuisine and can deter-

Characteristic	Social	Individual	Content
Heterogeneous		May transfer poorly to unseen items	Difficult to engineer / maintain
High risk	Not sufficiently accurate / reliable	Requirements and constraints usually needed	Domain constraints needed
High churn	Sparse data		
Implicit		Detailed requirements not available	
Unstable preferences	Historical data unreliable	Historical data unreliable Need user requirements	
Explanations needed	Explanations weak	Requirements can be mapped to items	Domain knowledge can be used

Table 1 Impact of recommendation domain on knowledge sources

mine that a Thai restaurant would be more similar to a Vietnamese restaurant than a German one would be.

Many high risk choices have constraints imposed by the domain that a recommender needs to obey. For example, a recommender for financial products [39] may know that certain investment instruments are only suitable for customers with certain characteristics – a particular life insurance policy might not be available to persons over the age of 55, for example. The recommendation problem can be in some cases formulated entirely as constraint satisfaction with constraints being contributed both by the user and by the system.

A final category of domain knowledge is means-ends knowledge, which is the knowledge that enables a system to map between the user's goals (ends) and the products that might satisfy them (means). For example, a camera buyer might not know much about digital cameras, but he might know that he wants to take photos of his daughter's basketball games. Part of the reason that users benefit from recommender systems is that they can make good choices without necessarily being conversant with all of the complexities of the product space.

Table 1 summarizes these domain considerations and their impact on knowledge sources.

6 Mapping Domains to Technologies

Some basic considerations come to the fore in considering the recommendation domain. First, there are some domain types for which social knowledge seems not very useful, in particular, high risk domains and ones with high churn. In high churn domains, there may not be enough time for an item to build up a reputation among a large number of peer users before it is replaced with other items.

Factor		Collaborative	Content-Based	Knowledge-Based
Heterogeneous	Low	✓	✓	✓
	High	+	−	−
Risk	Low	✓	✓	✓
	High	−	−	+
Churn	Low	✓	✓	✓
	High	−	+	+
Interaction style	Implicit	+	+	−
	Explicit	✓	✓	+
Preferences	Stable	✓	✓	✓
	Unstable	−	−	+
Scrutability	Required	−	−	+
	Not Required	✓	✓	✓

Table 2 Domain factors and recommendation techniques

When there is large risk associated with a domain, most users are going to need a more convincing explanation of the appropriateness of a recommendation beyond simply that others liked it. This is particularly important if we consider the problem of robustness in collaborative systems discussed elsewhere in this volume. Even a profile learned from the user's previous interactions might not be acceptable if adherence to it overrode considerations crucial to the current context.

Similarly, if we look at the interaction, we can see that it is not always possible to gather every kind of knowledge type from every type of interaction. In systems with implicit inputs, we do not gather any kind of direct requirements from the user (although it is sometimes possible to extract an implicit query from the user's activity with other applications, as done in the Watson system [40].)

Preference instability favors knowledge-based techniques. Learning over a user's prior interactions may turn out to be a hinderance rather than a help. However, in certain cases, such as web personalization, users may provide enough implicit data in a single session to form a useful profile that can be compared to others.

Table 2 shows the influence of the different domain factors on the choice of recommendation approach.

In cases where the criteria do not help to reach a definitive conclusion, it is worth noting that the different technologies do have different implementation and maintenance costs. Collaborative recommendation is likely to be the least expensive to implement. It requires a database of user ratings, but it does not require clean, well-engineered item features, which is the minimum requirement for the other recommendation technologies. Knowledge-based technologies are going to be the most expensive approach requiring knowledge engineering and continuing maintenance. So, a developer might wish to start by implementing the least expensive solution compatible with the domain.

Another factor to consider is that with hybrid recommendation it is possible to combine techniques. For example, to deal with a heterogeneous environment with unstable preferences, a hybrid between content-based and collaborative recommendation may be desirable. See the chapter on hybrid recommendation in this volume for more details.

6.1 Algorithms

If a domain can be clearly characterized as appropriate for one recommendation technology or another, a natural next question is which algorithms are appropriate? A thorough treatment of this question is beyond the scope of this chapter. Readers should review the relevant chapters of this volume. In the case of collaborative recommendation, it is possible to put forward some considerations.

In user-based collaborative recommendation, a subset of appropriate users are chosen based on their similarity to the active user , and a weighted aggregate of their ratings is used to generate predictions for the active user at run-time. Different implementations of collaborative filtering apply variations of the neighborhood-based prediction algorithms. Herlocker et al. [41] presents an empirical analysis of design choices in such algorithms and analyzes the variations of similarity metrics, weighting approaches, combination measures, and rating normalization.

Item-based collaborative filtering is a memory-based algorithm which explores the relationship between items as a function of how users have rated them. The item-based version of *kNN* algorithm has been shown to scale better and produce more accurate recommendation than user-based for large item collections [42].

Memory-based nearest-neighbor algorithms have two important computational limits: scale and sparsity. The need to compare each user against every other (n^2 comparisons) makes these techniques impractical for large collections. Also, the need to directly compare item ratings means that in very sparse collections, users may have very few neighbors.

In some databases, overall sparsity may hide the fact that there are dense sub-regions of the item space. Exponential popularity curves may make it possible to employ memory-based techniques because it is possible to find agreement among people or items in the dense sub-region and use that agreement to recommend in the sparse space [43]. (Jester [44] does this directly by creating a highly dense region of jokes rated by all users).

Dimensionality reduction (by way of singular value decomposition, latent semantic analysis, or other techniques) is by now a standard approach for coping with sparsity in ratings databases. [45, 46]. Various forms of compression and/or dimensionality reduction usually require extensive off-line computation, but as a result scale much better. The movie rating data released by Netflix prize which was also used for KDD cup competition in 2007 is an example of large, sparse data which motivated many research groups to develop new model-based algorithms [47, 48].

Other model-based collaborative algorithms include different machine learning techniques such as Bayesian networks [49] , and clustering [49, 50]. Bayesian networks are more practical for domains with high user preferences stability so that the user preference changes slowly with respect to the time needed to build the model.

Clustering techniques identify groups of users who appear to have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster[42]. Clustering techniques usually produce less-personal recommendations than other methods, and in some cases, the clusters have worse accuracy than nearest neighbor algorithms [49]. Clustering techniques can also be applied as a first step for shrinking the candidate set in a nearest neighbor algorithm or for distributing nearest neighbor computation across several recommender engines. While dividing the users into clusters may reduce the accuracy of recommendations, pre-clustering may be a worthwhile trade-off between accuracy and throughput [42].

6.2 Sample Recommendation Domains

Table 3 illustrates the application of these criteria in 10 different domains where recommendation applications exist. Not all combinations of the six criteria are represented, but we can see that the considerations given above are fairly predictive. High-risk domains generally lead to knowledge-based recommendation; scrutability is also a good predictor of this. Heterogeneous domains are handled largely with collaborative recommendation. Web page recommendation looks a bit contradictory when we consider high churn and preference instability, which would seem to militate against collaborative methods. However, as discussed above, database size can compensate for preference instability and these recommenders do collect large amounts of implicit preference data in each session. Also, heterogeneity is high, which argues in favor of using social knowledge.

7 Conclusion

This chapter considers recommender systems as intelligent systems, and as such, dependent on knowledge. The differences between recommendation approaches can be best understood through reference to the different knowledge sources that they employ. By considering how domain characteristics impact the availability and quality of knowledge sources, we can connect recommendation technologies and domain characteristics.

We have examined 6 different factors: heterogeneity, risk, churn, preference stability, interaction style, and scrutability, and considered their impact on the knowledge sources available for recommendation. From this analysis, we derive constraints on what recommendation technologies will be most appropriate for domains according to their characteristics. Application of these criteria to some existing systems shows that they do a reasonably good job of predicting what technologies have been successfully employed both in research and applications.

Domain	Risk	Churn	Heterog-eneous	Preferences	Interaction Style	Scrutabi-lity	Examples	Technology
News	Low	High	Low	Stable?	Implicit	Not re-quired	Yahoo news[51] ACR news[52] and [53] Google news[27]	Content-based Collaborative-Filtering
E-commerce	Low	High	High	Stable	Implicit	Not re-quired	Amazon.com eBay	Collaborative-Filtering
Web Page Recom-mender	Low	High	High	Unstable	Implicit	Not re-quired	[54, 55, 56]	Collaborative-Filtering Hybrid
Movie	Low	Low	Low	Stable	Implicit	Not re-quired	Netflix[57, 58] Movielens[59]	Collaborative-Filtering
Music	Low	Low	Low	Stable?	Implicit	Not re-quired	Pandora and [60, 61, 62]	Content-based Hybrid
Financial-services Life-insurance	High	Low	Low	Stable	Explicit	Required	Koba4MS[63] FSAdvisor[39] [64]	Knowledge-Based
Software En-gineering	Low	Low	Low	Stable	Explicit /Implicit	Required	[65] and [66]	Hybrid and Content-based
Tourism	High	Low	Low	Unstable	Explicit	Required	Travel Recom-mender [67] [68]	Content-based Knowledge-based
Job search Recruiting	High	Low	Low	Stable	Explicit	Required	CASPER [69] and [70]	Content-based
Real Estate	High	Low	Low	Stable	Explicit	Required	RentMe [71] FlatFinder[72] and [73]	Knowledge-based

Table 3 Sample domains for recommendation

Acknowledgements

This article is based on research performed by Ms. Ramezani at IBM Watson Research Center during the summer of 2007. An abbreviated version of the article with additional authors Lawrence Bergman, Rich Thompson and Bamshad Mobasher appeared as “Selecting and Applying Recommendation Technologies” at the Workshop on Recommendation and Collaboration at the Intelligent User Interfaces conference 2008.

References

1. R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

2. P. Resnick and H. R. Varian, “Recommender systems,” *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.
3. J. B. Schafer, J. A. Konstan, and J. Riedl, “E-commerce recommendation applications,” *Data Mining and Knowledge Discovery*, vol. 5, no. 1-2, pp. 115–153, 2001.
4. M. Montaner, B. López, and J. L. D. L. Rosa, “A taxonomy of recommender agents on theinternet,” *Artif. Intell. Rev.*, vol. 19, no. 4, pp. 285–330, 2003.
5. G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.
6. A. Felfernig and R. Burke, “Constraint-based recommender systems: technologies and research issues,” in *ICEC ’08: Proceedings of the 10th international conference on Electronic commerce*, (New York, NY, USA), pp. 1–10, ACM, 2008.
7. B. Smyth, E. Balfe, J. Freyne, P. Briggs, M. Coyle, and O. Boydell, “Exploiting query repetition and regularity in an adaptive community-based web search engine,” *User Modeling and User-Adapted Interaction*, vol. 14, no. 5, pp. 383–423, 2005.
8. R. Burke, “Interactive critiquing for catalog navigation in e-commerce,” *Artif. Intell. Rev.*, vol. 18, no. 3-4, pp. 245–267, 2002.
9. G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, “Incorporating contextual information in recommender systems using a multidimensional approach,” *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, 2005.
10. C. Rack, S. Arbanowski, and S. Steglich, *A Generic Multipurpose recommender System for Contextual Recommendations*, pp. 445–450. Washington, DC, USA: IEEE Computer Society, 2007.
11. A. I. Kovacs and H. Ueno, “Recommending in context: A spreading activation model that is independent of the type of recommender system and its contents,” in *Proceedings of Workshop on Web Personalisation, Recommender Systems and Intelligent User Interfaces In conjunction with AH 2006:International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (G. Uchyigit, ed.), (Dublin, Ireland), June 20-23 2006.
12. S. Berkovsky, L. Aroyo, D. Heckmann, G.-J. Houben, A. Kröner, T. Kuflik, and F. Ricci, “Providing context-aware personalization through cross-context reasoning of user modeling data,” in *UbiDeUM’2007 - International Workshop on Ubiquitous and Decentralized User Modeling, at User Modeling 2007, 11th International Conference, UM 2007, Corfu, Greece, June 26, 2007, Proceedings* (S. Berkovsky, K. Cheverst, P. Dolog, D. Heckmann, T. Kuflik, P. Mylonas, J. Picault, and J. Vassileva, eds.), 2007.
13. M. van Setten, S. Pokraev, and J. Koolwaaij, “Context-aware recommendations in the mobile tourist application compass,” in *Adaptive Hypermedia 2004* (W. Nejdl and P. De Bra, eds.), pp. 235–244, Springer Verlag, 26-29 August 2004, Eindhoven, The Netherlands 2004.
14. Z. Yu, X. Zhou, D. Zhang, C.-Y. Chin, X. Wang, and J. Men, “Supporting context-aware media recommendations for smart phones,” *Pervasive Computing*, vol. 5, no. 3, pp. 68–75, 2006.
15. M. Salam, J. Reilly, L. McGinty, and B. Smyth, “Knowledge discovery from user preferences in conversational recommendation,” in *Knowledge Discovery in Databases: PKDD 2005*, pp. 228–239, Springer Berlin / Heidelberg, 2005.
16. M. Zanker, “A collaborative constraint-based meta-level recommender,” in *RecSys ’08: Proceedings of the 2008 ACM conference on Recommender systems*, (New York, NY, USA), pp. 139–146, ACM, 2008.
17. D. McSherry, “Explanation in recommender systems,” *Artif. Intell. Rev.*, vol. 24, no. 2, pp. 179–197, 2005.
18. D. McSherry, “Explaining the pros and cons of conclusions in cbr,” in *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04)* (P. A. G. Calero and P. Funk, eds.), pp. 317–330, Springer, 2004. Madrid, Spain.
19. T. R. Roth-Berghofer, “Explanations and case-based reasoning: Foundational issues,” in *AAdvances in Case-Based Reasoning*, pp. 389–403, Springer Verlag, 2004.
20. J. L. Herlocker, J. A. Konstan, and J. Riedl, “Explaining collaborative filtering recommendations,” in *CSCW ’00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pp. 241–250, ACM Press, 2000.

21. S. Niwa, T. Doi, and S. Honiden, “Web page recommender system based on folksonomy mining for itng 06,” in *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on*, pp. 388–393, 2006.
22. C. Hayes, P. Avesani, and S. Veeramachaneni, “An analysis of the use of tags in a blog recommender system,” in *IJCAI-07, the International Joint Conference on Artificial Intelligence* (M. M. Veloso, ed.), pp. 2772–2777, 2007.
23. Z. Xu, Y. Fu, J. Mao, and D. Su, “Towards the semantic web: Collaborative tag suggestions,” in *Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006*, (Edinburgh, Scotland), 2006.
24. B. Markines, L. Stoilova, and F. Menczer, “Bookmark hierarchies and collaborative recommendation,” in *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*, AAAI Press, 2006.
25. H. Wu, M. Zubair, and K. Maly, “Harvesting social knowledge from folksonomies,” in *HYPERTEXT ’06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pp. 111–114, ACM Press, 2006.
26. S. Aciar, D. Zhang, S. Simoff, and J. Debenham, “Informed recommender agent: Utilizing consumer product reviews through text mining,” in *WI-IATW ’06: Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, (Hong Kong), pp. 37–40, IEEE Computer Society, 2006.
27. A. S. Das, M. Datar, A. Garg, and S. Rajaram, “Google news personalization: scalable online collaborative filtering,” in *WWW ’07: Proceedings of the 16th international conference on World Wide Web*, (New York, NY, USA), pp. 271–280, ACM, 2007.
28. B. Krulwich, “Lifestyle finder: Intelligent user profiling using large-scale demographic data,” *Artificial Intelligence Magazine*, vol. 18, no. 2, 1997.
29. M. J. Pazzani, “A framework for collaborative, content-based and demographic filtering,” *Artif. Intell. Rev.*, vol. 13, no. 5-6, pp. 393–408, 1999.
30. K. Lang, “Newsweeder: Learning to filter netnews,” in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339, 1995.
31. R. J. Mooney and L. Roy, “Content-based book recommending using learning for text categorization,” in *DL ’00: Proceedings of the fifth ACM conference on Digital libraries*, pp. 195–204, ACM Press, 2000.
32. M. Pazzani and D. Billsus, “Learning and revising user profiles: The identification ofinteresting web sites,” *Machine Learning: Special issue on multistrategy learning*, vol. 27, no. 3, pp. 313–331, 1997.
33. D. McSherry and D. W. Aha, “Mixed-initiative relaxation of constraints in critiquing dialogues,” in *ICCBR ’07: Proceedings of the 7th international conference on Case-Based Reasoning*, vol. 4626, pp. 107–121, 2007.
34. L. Chen and P. Pu, “Survey of preference elicitation methods,” tech. rep., Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland, August 2004.
35. D. Mladenic and M. Grobelnik, “Feature selection for unbalanced class distribution and naive bayes,” in *ICML ’99: Proceedings of the Sixteenth International Conference on Machine Learning*, (San Francisco, CA, USA), pp. 258–267, Morgan Kaufmann Publishers Inc., 1999.
36. R. Kosala and H. Blockeel, “Web mining research: a survey,” *SIGKDD Explor. Newsl.*, vol. 2, no. 1, pp. 1–15, 2000.
37. A. G. Hauptmann, “Integrating and using large databases of text, images, video, and audio,” *Intelligent Systems and Their Applications, IEEE*, vol. 14, no. 5, pp. 34 – 35, 1999.
38. O. R. Zaïane, J. Han, Z.-N. Li, S. H. Chee, and J. Y. Chiang, “Multimediaminer: a system prototype for multimedia data mining,” in *SIGMOD ’98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pp. 581–583, ACM, 1998.
39. A. Felfernig and A. Kiener, “Knowledge-based interactive selling of financial services with fsadvisor,” in *Proceedings of the National Conference on Artificial Intelligence*, pp. 1475–1482, 2005.
40. J. Budzik, K. Hammond, and L. Birnbaum, “Information access in context,” *Knowledge based systems*, vol. 14, no. 1-2, pp. 37–53, 2001.

41. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering,” in *SIGIR ’99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, NY, USA), pp. 230–237, ACM Press, 1999.
42. B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in *WWW ’01: Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, ACM, 2001.
43. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, “Evaluating collaborative filtering recommender systems,” *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.
44. K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, “Eigentaste: A constant time collaborative filtering algorithm,” *Inf. Retr.*, vol. 4, pp. 133–151, July 2001.
45. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Application of dimensionality reduction in recommender systems-a case study,” in *Proceedings of ACM WebKDD Workshop*, 2000.
46. S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman, “Using singular value decomposition approximation for collaborative filtering,” in *CEC ’05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, (Washington, DC, USA), pp. 257–264, IEEE Computer Society, 2005.
47. S. Rosset, C. Perlich, and Y. Liu, “Making the most of your data: Kdd cup 2007 ”how many ratings” winner’s report,” *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 66–69, 2007.
48. M. Kurucz, A. A. Benczúr, T. Kiss, I. Nagy, A. Szabó, and B. Torma, “Kdd cup 2007 task 1 winner report,” *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 53–56, 2007.
49. J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the 14th Conference on Uncertainty in Artiffcial Intelligence*, pp. 43–52, 1998.
50. L. Ungar, D. Foster, E. Andre, S. Wars, F. S. Wars, D. S. Wars, and J. H. Whispers, “Clustering methods for collaborative filtering,” in *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence*, AAAI Press, 1998.
51. D. Billsus and M. J. Pazzani, “User modeling for adaptive news access,” *User Modeling and User-Adapted Interaction*, vol. 10, no. 2-3, pp. 147–180, 2000.
52. B. Mobasher, R. Cooley, and J. Srivastava, “Automatic personalization based on web usage mining,” *Communications of the ACM*, vol. 43, no. 8, pp. 142–151, 2000.
53. V. Maidel, P. Shoval, B. Shapira, and M. Taieb-Maimon, “Evaluation of an ontology-content based filtering method for a personalized newspaper,” in *RecSys ’08: Proceedings of the 2008 ACM conference on Recommender systems*, pp. 91–98, ACM, 2008.
54. P. Buono, M. F. Costabile, T. Guida, and A. Piccinno, “Integrating user data and collaborative filtering in a web recommendation system,” in *Lecture Notes in Computer Science: Hypermedia: Openness, Structural Awareness, and Adaptivity*, pp. 192–196, SpringerLink, 2002.
55. J. Li and O. R. Zaiane, “Combining usage, content, and structure data to improve web site recommendation,” *Lecture Notes in Computer Science : E-Commerce and Web Technologies*, pp. 305–315, 2004.
56. S. S. Anand and B. Mobasher, “Introduction to intelligent techniques for web personalization,” *ACM Trans. Interet Technol.*, vol. 7, no. 4, p. 18, 2007.
57. A. Paterek, “Improving regularized singular value decomposition for collaborative filtering,” in *Proc. of the of the KDD Cup and Workshop 2007 (KDD 2007)*, August 2007.
58. G. Takács, I. Pilászy, B. Németh, and D. Tikk, “On the gravity recommendation system,” in *Proc. of the of the KDD Cup and Workshop 2007 (KDD 2007)*, pp. 22–30, August 2007.
59. N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl, “Combining collaborative filtering with personal agents for better recommendations,” in *AAAI:Conference on Artificial Intelligence*, pp. 439–446, 1999.
60. C. Hayes and P. Cunningham, “Smart radio: Building music radio on the fly,” in *Proceedings of Expert Systems 2000, Cambridge, UK*, 2000.
61. Y. Hijikata, K. Iwahama, and S. Nishida, “Content-based music filtering system with editable user profile,” in *SAC ’06: Proceedings of the 2006 ACM symposium on Applied computing*, pp. 1050–1057, ACM, 2006.

62. H.-C. Chen and A. L. P. Chen, "A music recommendation system based on music data grouping and user interests," in *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pp. 231–238, ACM, 2001.
63. A. Felfernig, "Koba4ms: Selling complex products and services using knowledge-based recommender technologies," in *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pp. 92–100, IEEE Computer Society, 2005.
64. A. Tartakovski, M. Schaaf, and R. Bergmann, "Retrieval and configuration of life insurance policies," in *Lecture Notes in Computer Science, Case-Based Reasoning Research and Development*, pp. 552–565, Springer Berlin / Heidelberg, 2005.
65. C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, "Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes," in *RE '08: Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*, (Washington, DC, USA), pp. 165–168, IEEE Computer Society, 2008.
66. R. Holmes, R. J. Walker, and G. C. Murphy, "Approximate structural context matching: An approach to recommend relevant examples," *IEEE Transactions on Software Engineering*, vol. 32, no. 12, pp. 952–970, 2006.
67. F. Ricci, "Travel recommender systems," in *IEEE Intelligent Systems*, pp. 55–57, November/December 2002.
68. T. Mahmood, F. Ricci, A. Venturini, and W. Hpken, "Adaptive recommender systems for travel planning.," in *Information and Communication Technologies in Tourism 2008* (P. O'Connor, W. Hpken, and U. Gretzel, eds.), pp. 1–11, Springer, 2008.
69. D. H. Lee and P. Brusilovsky, "Fighting information overflow with personalized comprehensive information access: A proactive job recommender," in *ICAS '07: Proceedings of the Third International Conference on Autonomic and Autonomous Systems*, (Washington, DC, USA), p. 21, IEEE Computer Society, 2007.
70. J. Malinowski, T. Keim, O. Wendt, and T. Weitzel, "Matching people and jobs: A bilateral recommendation approach," in *HICSS '06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, pp. 137c–137c, IEEE Computer Society, 2006.
71. R. Burke, "Knowledge-based recommender systems," in *Encyclopedia of Library and Information Systems*, Marcel Dekker, 2000.
72. P. Viappiani, P. Pu, and B. Faltings, "Conversational recommenders with adaptive suggestions," in *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pp. 89–96, ACM, 2007.
73. J. Zhang and P. Pu, "A comparative study of compound critique generation in conversational recommender systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4018 NCS, pp. 234–243, Springer Verlag, Heidelberg, D-69121, Germany, 2006.