How to get the Recommender out of the Lab?

Jérôme Picault[1], Myriam Ribière[2], David Bonnefoy[3], and Kevin Mercer[4]

1 Introduction

A personalised system is a complex piece of software made of many interacting parts, from data ingestion to presenting the results to the users. A plethora of methods, tools, algorithms and approaches exist for each piece of such a system: many data and metadata processing methods, many user models, many filtering techniques, many accuracy metrics, many personalisation levels... In addition, a real-world recommender is a piece of an even larger and more complex environment over which there is little control: often it is part of a larger application introducing constraints for the design of the recommender, e.g. the data may not be in a suitable format, or the environment may impose some architectural or privacy constraints. This can make the task of building such a recommender system daunting.

This chapter intends to be a guide to the design, implementation and evaluation of personalised systems. It will present the different aspects that must be studied before the design is even started, and how to avoid pitfalls, in a hands-on approach.

2 Designing real-world recommender systems

Previous work in the literature provides guidelines on many aspects of building a recommender system. For example, [48] lists some characteristics and general principles that should drive a personalised system design, such as taking into account content specificity, importance of trust in the system and of involving users. [25] provides an extensive analysis of methods and metrics for evaluating collaborative filtering systems, including also a taxonomy of user tasks for recommender systems,

[1]Alcatel-Lucent Bell Labs, e-mail: jerome.picault@alcatel-lucent.com ·[2]Alcatel-Lucent Bell Labs, e-mail: myriam.ribiere@alcatel-lucent.com ·[3]pearltrees, e-mail: david.bonnefoy@pearltrees.com ·[4]Loughborough University, e-mail: K.C.Mercer@lboro.ac.uk

and an interesting description of dataset properties. This work is extremely useful once initial technological choices have been made (user model, choice of algorithm, etc.). But *how can we make sensible choices when initially designing the system?* This is a major concern as any change later in the development is costly.

In order to tackle this problem in a systematic way, it is useful to step back and see from a wider perspective what are the main design decisions to make and the factors which influence them. Fig. 1 illustrates the approach suggested in this chapter.

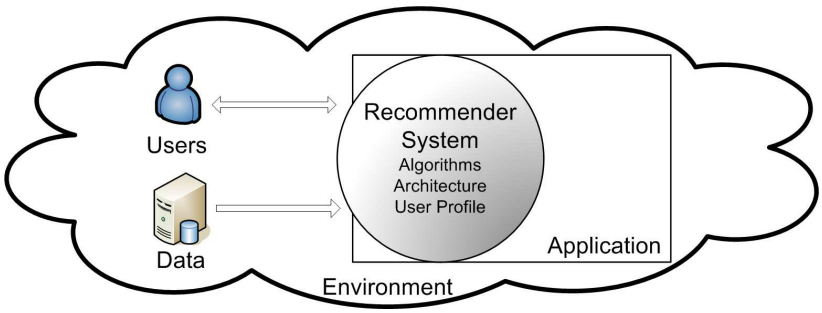

Fig. 1: The Recommender in its environment

Designing a recommender system means making choices that can be categorised into the following domains:

- Algorithms: which recommendation methods to use ;
- Architecture: how will the system be deployed, will it be centralised or distributed?
- User profile: what is the user model, is profile adaptation needed?

For a large part, these choices are constrained by the environment of the recommender. It is thus important to systematically study the environment the system will be situated in. We propose to describe it along three dimensions:

- Users: who are the users, what are their goals?
- Data: what are the characteristics of the data on which recommendations are based?
- Application: what is the overall application the recommender is part of?

We propose to build a model of the environment based on these three dimensions, and base the recommender system design on these models.

The following sections of this chapter will describe this process. The next section first describes the three models that should be built prior to the system design and how they affect the design decisions. In section 4 we will then show how these models can help in evaluating the system. The section 5 will finally present a use-case of the methodology.

3 Understanding the recommender environment

As mentioned in the previous section, we propose to define three models (user, data, and application). These models will assist the recommender designer in decision making processes, helping them understand the key constraints of their future system, ask themselves the right questions and define constraints for making decisions about three main aspects: choice of the recommendation algorithm, choice about the recommender system architecture and choice in the possible adaptation of the user profile. Our methodology to define the environment models consists in defining key aspects of each model and the key questions to be asked throughout the process.

3.1 Application model

Though a recommender system is itself a complex piece of software, it is by nature part of a larger system. A recommender is one of the features of an overall application. It may be a minor feature or a main selling point; the application may be pre-existing or built together with the recommender, but in any case the design of a recommender system has to be integrated within the design of the application hosting it. This section studies the main factors regarding the host application that should influence the recommender design along two main lines: the role of the recommender and the influence of the application implementation (Table 1).

Table 1: Application model

Model's features	**Possible values**
Recommender purpose	main service, long-tail focused, increase revenues, increase loyalty, increase system efficiency
Recommender type	single item, multiple items, sequence
Integration with navigation features	yes, no
Performance criteria (correctness, transparency, serendipity, risk-taking, response speed, robustness to attack)	performance target on each criterion
Device to support the application	fixed, mobile, multiple
Number of users	single, group
Application infrastructure	browser-based application, distributed application
Screen real-estate	limited, not limited

3.1.1 Understanding the recommender role in the application

The main question to solve before designing a recommender system is to be very clear on its goals within the overall application. This is often not as easy as it seems, and can have fundamental consequences on the type of system to be built. Two perspectives have to be studied: the application point of view and the user point of view. They overlap, but are still separate. The user point of view is studied in detail in section 3.2. This section focuses on the application point of view.

Purpose of the recommender: From the application side, a recommender system may have different purposes, for instance:

- It may be a *major service* provided by the application. Many such recommender systems have been developed in different fields such as music (Pandora[1], last.fm[2], MyStrands[3] ...) or movies (MovieLens[4] , Netflix[5] ...)
- To *take advantage of the 'Long Tail'*, as first described by Chris Anderson in [5]. The idea that recommendations can give easy access to previously hard to find items is central to the business model of many e-commerce web sites. In that case, the recommendations have to be focused on lesser known items, accurately tailored to each user.
- To *increase loyalty* of users: customers return to the services that best match their needs. Loyalty can be increased by involving users in the recommendation process (ask for ratings or manual profile, highlight new recommendations, etc.)
- To *increase revenues* through the promotion of targeted products. In that case, the recommendations would be determined both by the user preferences and some marketing rules defined to follow a particular strategy. It is necessary to carefully balance the expectations of the users and the business strategy, to ensure users perceive value in the system.
- To *increase system efficiency*. By allowing the user to more directly get the content he is looking for, a recommender system can lower the amount of data to be exchanged, thus lowering the costs of running a system.

Recommendation type: a recommender can provide several sorts of recommendations, from a single item (or a simple list of items) to a sequence (e.g. in a travel recommender system). Single item or simple list recommenders do not take into account how the choice of an item by the user at a given point of time may influence the choice of next items. The choice of the recommendation type may be driven by the need or presence of a logical order in the recommendations. For example, in a travel recommender system, a trip can be seen as a sequence of travel steps (such as visiting a museum, going to the beach, etc.), which can be connected through various logical features, such as geography, culture, history, leisure, etc. in order to provide a good travel experience. Recommendations of sequences of items may be particularly useful when users are new to a domain and need a path in the selection of diverse items, helping them to go through their personal development goals: the logical order of the recommendations help them progressing in their learning curve by providing the next most appropriate step(s). Unfortunately, there is to date little work on recommenders of sequences. Some of our ongoing work is addressing this issue; other techniques coming from data mining domain, such as the Apriori algorithm [2], may be used.

Integration with content navigation features: Another key point to study is how the recommendations will integrate with other content navigation features. In

[1] Pandora Internet Radio : `http://www.pandora.com`

[2] Last FM : `http://www.last.fm`

[3] MyStrands, Social Recommendation and Discovery : `http://www.mystrands.com/`

[4] MovieLens, Movies Recommendations : `http://www.movielens.org/`

[5] Netflix: `http://www.netflix.com/`

most cases, users will be offered other means to browse content in addition to getting recommendations. A good integration of these different navigation methods can greatly enhance the user experience.

- Users may request recommendations completely separately from content browsing. This can be a good choice if recommendations are to be highlighted as a main feature of the application. Such recommendations may also appear on the home page of a web site or home screen of an application.
- It can also be beneficial to have recommendations dependant on the current interaction context. The typical case is to recommend items that are similar to those the user is currently browsing. In that case, the recommender system must be able to provide recommendations tailored to the context, e.g. the current genre when browsing music.
- It is also important to consider whether the use of the recommender system is optional or a mandatory part of the interaction model. This has strong implications on the expected reliability of the system: failure to complete a major task on a website because the only way of completing that task was using a recommender system which offered inaccurate recommendations could be a source of major user dissatisfaction. However within a system design where the recommender sat in parallel to more tradition navigation methods, the impact of the same recommender may be many times less severe.

Performance criteria: Once these goals are clarified, it is possible to define targets for the performance of the system along a number of criteria. Not only these criteria will allow evaluating the system once it is built, but they are also key to selecting the proper algorithms. Many criteria can be used, see [25] for a comprehensive reference of many possible criteria. Some key ones could include:

- *Correctness metrics*, such as accuracy, precision and recall: these are the technical criteria that can be used to evaluate recommendation algorithms, and have been the focus of many studies over the years. However, they are actually not sufficient to evaluate user satisfaction [33].
- *Transparency and explanability*: how important is it that users understand how the recommendations have been determined? A good level of transparency can be more difficult to achieve with some families of algorithms. For instance, collaborative filtering offers little transparency naturally, but [24] proposes an analysis of the problem and some solutions.
- *Serendipity*: should users be (hopefully pleasantly) surprised by some of the recommendations or is it desirable to allow obvious recommendations? Canonical collaborative filtering tends to recommend items that are very popular, and may be considered obvious and of little use to most users. Techniques exist to correct this tendency; some are described in [25].
- *Risk taking*: related to the previous criterion, should the recommendations be made only for items that the user has a high probability of liking? More risky items can be recommended if the goal is to allow the user to discover content they would not be aware of without the help of the system.
- *Response speed / performance*: in many cases, the reactivity of the application is a major concern and can be sometimes more important than the accuracy of the

results. Knowing how many recommendations are needed per time unit allows to better choose algorithms or decide if recommendations should be precomputed.
- *Reliability*: What is the criticality of the recommender output in the context of the given application? For instance the design of a recommender for an e-commerce website would not be approached in the same way as a solution for an organ donor matching system in a hospital.
- *Robustness* to attacks: in particular if the recommender system has a commercial role (for instance if it recommends products for purchase), it may be subject to attacks to skew the results. [34] presents a thorough analysis of possible attacks and some solutions for collaborative filtering algorithms.

3.1.2 Understanding the influence of the application implementation

In addition to the features seen by the users, some aspects of the application implementation also have a large influence on how the recommender can be designed.

Single or multiple devices: the same application may be accessed from a single or multiple devices (e.g. a news recommender system on mobile, PC, set-top box). It must be studied whether the recommendations should depend on the user context (see section 3.2). But in term of implementation, it also raises additional questions: should the collected preferences be merged or should they remain separate to enable contextual recommendations? Where should the preferences be stored? If the preferences are stored on a server, are they transmitted to the server in real-time (implying a constant connection) or in batches? Answering such questions is important even if access from multiple devices is not initially planned, as it is becoming the norm that web applications are derived into mobile versions.

Single or multiple users: conversely, the same device may be used by several users. The consequences of this user social environment are studied in section 3.2. In addition, users that interact with the personalised application may be either registered or anonymous and may interact frequently or occasionally. This impacts the architecture of the recommender (requires diffent identification means, e.g. login vs. cookies), algorithm choice (e.g. session profile in case of anonymous occasional users vs. persistent profile in case of registered users), and algorithm parameters (e.g. the degree of adaptability of the system to the user – i.e. the rapidity of profile adaptation: long-term vs. short-term – should depend on the frequency of use).

Application infrastructure: The infrastructure the application runs on puts strong constraints on the types of recommendation algorithms that can be used and on their specific implementation. In particular, the scalability of the solution has to be carefully studied. Two main cases can be identified, whether the application is accessed through a browser or if an application runs locally on the user device.
- *Browser-based application*. In the case of a browser-based application, the processing done on the client will be minimal. As all information is available at a single point, any kind of algorithm can be used. However, scalability will be a major focus in the design.

- *Distributed application*. When the user device runs a local application, different architectures may be used. To determine the most suitable distributed architecture, the following criteria must be studied:
  - Processing power of the relevant devices. Is the client device able to support intensive tasks? Can the server-side computing resources be extended as needed when the number of users grows?
  - Network connectivity. Is the network connection permanent? Does the data transfer have a cost for the users? For mobile devices, what is the impact of the connection to the battery life?
  - Data source. How are the data that the recommendations are drawn from accessed? Is it from a database (information retrieval) or a stream of data (information filtering)?

Content filtering algorithms may run entirely on the client device. This has several advantages: privacy is preserved as no personal information has to be transmitted; such an architecture is very scalable as the number of users has minimal impact on the server load. On the other hand, algorithms such as collaborative filtering require that information from all or large subsets of users be collated. In that case, a fully centralised architecture may be used with most of the same pros and cons as a browser-based application, but with the additional need for a mechanism to update the client application when needed. More sophisticated architectures can also be designed, such as in the case of the TV programme recommender TiVo [3]. In this system, a part of the computation is done server-side and another part is done on each set-top box. Other complex architectures include distributed profile management [13] or mechanisms to make different recommenders running on different devices communicate [28]. The choice of infrastructure may be influenced by other components the system should connect to such as external web services.

Screen real-estate: a point that is easily overlooked in the early stages of the design is how much screen space the recommendations will use. In many cases it is very limited and constrained by the application user interface design. This is not only a quantitative issue and can have influences on the very nature of the recommendations that are provided. For instance, if it is intended to provide more exploratory recommendations, it is necessary to have a sufficient number of recommendations and thus sufficient space. The same problem may arise for example in the display of a recommendation profile to a user; a solution to display the user profile on a small device has been proposed [40].

The study of the application the recommender system will be part of brings a first set of constraints on the design of the recommender, but on its own it is not enough to build an appropriate system. This additionally requires knowledge about both the user and the data.

3.2 User model

Fully understanding the user is a fundamental component to the success of any recommender system. Insights into the end users which are to be built into the user model must come early enough in the development lifecycle to influence major design decisions surrounding the selection of technology. Applying a user-centred approach to any project within the initial phases can greatly reduce the need for extensive redesign, maintenance and customer support [9, 22].

In this section, we propose to characterize users by a number of properties that may have an impact on the recommender system design and choices the designer will have to face (Table 2).

Table 2: User model

Model's features	**Possible values**
Demographics information	yes, no
Goal existence	yes, no
Goal nature	implicit, explicit
Level of expectation	high, medium, low
Handling change of expectation over time	yes, no
Limited capabilities of user device	yes, no
Importance of user situation	high, medium, low
Social environment	alone, with others
Trust and privacy concerns	high, medium, low

At a fundamental level the aspects of the user which must be understood when developing a recommender revolve around best practices for understanding and specifying the context of use for an interactive system in a human-centred way [26]: Who are the end users? What expectations and goals lie behind the users motivations to use the system the recommender supports? What are the user centric contextual factors surrounding use of the system? In fully answering each *context of use* question, fundamental requirements for the system design and technology choices will be uncovered.

3.2.1 Understanding who the users are

Understanding who the users of the recommender system will be should revolve around three main concerns: understanding their key identifying characteristics, their skill levels and their prior experience with similar systems. We concentrate on the identification of user characteristics because it has special utility in terms of recommender design.

Identifying User Characteristics: Gathering a picture of the different user groups through both demographic information such as age, gender, job area, nationalities, spoken languages, and deep qualitative insights from user research are an important jumping off point in the development of recommender user models. Understanding these factors allows the development team to start to build a relationship with the users and get an appreciation of their needs.

Development of user group clusters may allow (1) the building of simple recommenders based on demographics. This is commonly used in targeted advertising solutions to cluster customers into segments [23]; (2) define stereotypes of users [42]: stereotyping techniques allow the definition of a set of differentiating characteristics for a group of users; when a new user is introduced into the system, they can be assigned to a predefined stereotype, based on their personal data, which allows the activation of a set of default preferences that may be further refined over time thanks to user profile adaptation methods [17]. Personalisation solutions exploiting user characteristics can be used in combination with more sophisticated techniques to provide a first simple step in a hybrid filtering process, or to bootstrap a content based filtering algorithm by using stereotype profiles.

In addition, the type of user (e.g. professional users vs. end users) is an essential criterion to help determine the user's level of expectations and so, to choose algorithms accordingly.

3.2.2 Understanding users' motivations, goals and expectations

Goals and motivations: The designer of the recommender system needs to identify the user tasks [25] and understand if the application can support completion. For example the Amazon web site's offering to the user revolves around the possibility to buy items and get recommendations for items to buy. From the user's viewpoint, their motivation for using the service is to complete one of the two goals of either buying an item for themselves, or buying an item for someone else. The Amazon recommender however, does not differentiate those two goals and therefore provides inaccurate recommendation results in one of the use cases. As another example, a search engine coupled with a content recommender can offer the opportunity for the user to browse the Internet and find information according to a request. The user in this context may be motivated by the need to complete a specific targeted goal, or their motivation may be simply to relax and spend some time browsing for fun. Identifying and understanding user motivation can result in fundamental recommender and user experience improvements. User insights captured within the user model (Table 2) allow designers to consider the possible range of tasks the future application needs to support.

In most cases there will be many motivations for the use of a system, and a designer must consider ways of finding the nature of the goal, either explicitly or implicitly. An *explicit* goal may be either defined by the application (Amazon could have added a button asking the user if they were buying the item for themselves or someone else) or expressed by the user (for example through a set of queries). An *implicit* goal may be either predefined by the application itself (such as in personalised news pushing systems where the system provides stories to the user in the belief that the user goal is to feel more informed about particular world events), or can be inferred from the user's interactions. In contrast it is quite possible that a clear task goal is not discernable (e.g. because the application is dedicated to enjoyment). In such cases it can be difficult to build a good recommender as user satisfaction

may be more strongly related to external factors such as user mood outside of the system's control. In this case, spread, quantity or serendipity may be the most desirable qualities of the recommender.

The impact of the user's goal on filtering algorithms and diversity heuristics [54, 53] when displaying the results of a recommender is important and can generate user dissatisfaction if not addressed correctly at the design stage. For example a content-based method may be more adapted for focused people (because of the specialization of the results), whereas collaborative methods may be more adapted to people with less focused goals (because of the broader diversity of the results).

Users' expectations: The implicit or explicit goal of the user as described in the previous section is the key to evaluating the level of user expectation:

- *High expectation levels* would be seen in the context of goal-oriented users, who are focused on completing the current task. It means that the recommender must target an "all good items" [25] list of recommendations to achieve a high level of satisfaction for the user. This level of expectation is also linked to decisions made at the application level (see sec. 3.1), where the place of the recommender in the overall application and the constraints of the targeted device are central to the user expectations. Returning to the news pushing system as an example, that system needed to provide an "all good items" list. If the user cannot find the right personalised news in the first ten recommendations, they must at best start scrolling down to search through the content and at worst they reject the application never using it again because of their initial dissatisfaction at first use.
- *Medium expectation levels* can be seen as the recommender returning "some good items" [25]. If the user has choice and flexibility in the use of the recommender to complete their task, the user expectation is lowered. In this category we also find people that use the recommender purely to evaluate if the system corresponds to their expectation of what a recommender can bring to them. Some are looking for recommendations that are exactly aligned to their preferences; others would want to discover new suggestions that are different from their current habits.
- *Low expectation levels* are mainly a target for personalised applications used in an opportunistic context. As indicated by [45] on recommendation of web sites, "research has shown that if the task is intrinsic, i.e. just browsing for fun, it is very difficult to recommend sites".

Each level of expectation leads to various attitudes from end users driven from the level of dissatisfaction, from deleting the application, to using it only for fun.

Handling changes to expectations over time: When using a recommender system, users' expectations may change over time. This is important to consider with respect to the need to develop an adaptive recommender or not. The performance of most recommender systems evolves over time; with increases in user profile information comes better accuracy. Users too have expectations of a system's capabilities at first use and these also change over time as both familiarity with the system increases and their own needs change. It is not only important that recommendation systems can adapt to the needs of users over time, but also that they can demonstrate system performance early enough in the use lifecycle to match the user's ex-

pectations, building trust in the recommender output and ensuring continued user engagement.

3.2.3 Understanding users' context

The final area to consider are contextual issues surrounding use of the recommender:

User's device: The first consideration is what device will the user use to access the recommender? Recommenders are now a prevalent feature on applications and services accessible on devices as diverse as mobile handsets, desktop PCs, and set top boxes. The implications of this on the system design are studied in section 3.1.

Situation of interaction: Situational considerations include firstly location, as the advent of ubiquitous mobile use has now made this a prime consideration. Location considerations may cover absolute geography such as if you wished to recommend attractions in the local area on a mobile device and also relate to understanding user activities beyond absolute positioning, e.g. contexts such as is the user at work, or are they shopping? Another important contextual consideration is temporal factors. It may be important for the recommender system to consider time information within user profile data collection when modelling the user. As an example, a user may watch news on mobile TV on the train into work, but prefers to watch comedy on the same train on the way home. Some solutions have been proposed to take into account these situations in the recommender model; for example with an explicit link between the interests and a given situation [12], or alternatively, the link between preferences and context information can be done implicitly [29]. The combination of location and temporal information in data collection can assist greatly in building a more robust recommender system [4].

Social environment: An important consideration here is whether the use of the system is usually carried out alone or with other people. This affects many design decisions including algorithm choice, data collection methods and recommendation presentation: e.g. this question helps decisions regarding strategies to provide group recommendations, for example merging of individual preferences to obtain a unique group profile to be used in the content retrieval process [6, 31, 52], or the application of a consensus mechanism by users in order to cooperatively define a shared content retrieval policy [27, 32] or other methods described in [15]. However even in a group, people may still require individual recommendations, as explained for TV in [8]; for this example, some solutions have been proposed in [11, 30].

Consideration of all these questions requires considerable effort and raises the possibility for extensive user research. Though this may seem a very large activity, understanding which of these areas is relevant to the recommender under development can refine the amount of research needed.

3.3 Data model

The last point the designer should study carefully are the characteristics of the items the system will exploit and manipulate. Indeed, item descriptions generally pre-exist before the personalised system, and the designer of the recommender system has little possibility to influence or change them. We propose a data model, helping to identify the main characteristics of data that may influence the design and the results of the future recommender. The designer shall implement our data model, i.e. for each feature of the data model, they have to think about the possible values presented in Table 3.

Table 3: Data model

Model's feature	**Possible values**
Data type	structured, semi-structured, unstructured
Metadata quality and quantity	high, medium, low
Metadata expressiveness	keyword-based, semantic-based
Description based on standards	yes, no
Volume of items	a lot, few
Diversity of items	homogeneous, heterogeneous
Distribution of items	long-tail, mainstream
Stability vs. persistence of items	stable, changing, changing a lot
User ratings	implicit, explicit, none
Type of rating	binary, multi-level...

3.3.1 Understanding the type of available data to describe items

There exist several ways to describe items:

- **unstructured data**: an item can be represented only by unstructured data, which refers to information that either does not have a data model or one that is not easily usable by a computer program (e.g. audio, video, unstructured text). In such cases, a pre-processing needs to be made to extract significant keywords, or concepts helping to distinguish each item from each other. The fact of having unstructured data is not blocking per se, because in many cases there are a number of techniques to obtain a set of metadata describing the items. For text analysis, tools such as GATE [19] or Lucene[6] allows the extraction of keywords from unstructured text. Techniques for extraction of metadata from other types of data such as multimedia content (images, videos) are less reliable though, and often require to combine them together.
- **semi-structured data**: an item is often described by several generic metadata, corresponding to the main characteristics of the item and free text. A typical example of semi-structured data is an Amazon[7] product page, or a TV programme, which is expressed with a number of properties such as programme genre, programme schedule, etc. Each property takes values in a finite set of vocabulary

[6] Lucene, An Open Source Information Retrieval Library, `http://lucene.apache.org/`

[7] Amazon, `http://www.amazon.com`

(that belong to a taxonomy or an ontology), or includes non-structured elements (e.g. the synopsis of the programme). In this case, the evaluation of the quantity and quality of data is required to know if the item metadata should be enhanced by performing some processing and analysis on the unstructured part.
- **structured data**: items can also already be described by a well structured model that could belong to a standard, or a de facto standard, such as TVAnytime[8] for the description of TV programs in an electronic program guide (EPG) for DVB-H. When having structured data, the designer must examine the quantity and quality of data to anticipate the potential lack of information for their use in the frame of a personalised system.

This first data feature has many impacts. First, on algorithms: if the data are unstructured and the cost to extract pertinent metadata is too high or too imprecise, a full set of recommender algorithm families are excluded: content-based approaches to recommendation [50], Bayesian model [21], rule-based system and distance metrics. Only collaborative filtering methods could be used in such a configuration. Second, on user models: data representation should be reflected in the user model: for example, if a piece of text is represented as a vector of keywords, then content based method will lead to the use of a vector of keywords representation of user profile. In the case of unstructured data, the user model can be very simple, such as the history of the user's content consumption.

3.3.2 Understanding the quality / quantity of metadata

Quality and quantity of structured data are important performance factors of a recommender system, especially when using a content-based recommender. Several properties of metadata play a significant role in the choices that can be made with respect to the design of the personalised system.

Quality: the designer has to have some clues about the quality of the metadata available and has to understand the actions to take in the case where metadata are of medium or low quality. In general, an item metadata description is considered as of high quality if it enables one item to be distinguished from another. For example in news RSS feeds, two items can share common keywords corresponding to a category of news (e.g. sport news, football), but must have sufficient additional keywords for distinguishing news related to a certain match, a global event like a competition, a football player etc. In that example, the designer has to understand the right level of details in the description to capture the differences between news feeds. The quality feature of metadata in our model is a major point of decision for the recommender designer. He has to balance the accuracy of recommendation results and the recommender performance in terms of time to respond, storage capacity and processing cost. For example, if the designer prefers best performance of recommendation, he would have to introduce some constraints on the architecture and avoid implementing the recommender on a lightweight client. The designer can

[8] TV Anytime, `http://www.tv-anytime.org/`

also choose to perform recommendations using a two step algorithm, distributed between the server and the client device [37]. This quality feature is also linked to the role of the recommender in the application and the expectation of users according to this role (see sec. 3.1 and 3.2).

Expressiveness: the future users are also fundamental to the process of metadata evaluation. Metadata must reflect the users' points of view on items, and represent what users consider as differentiating characteristics of items. So the *expressiveness* of metadata is crucial to the performance of the recommender. Items can be expressed with a large variety of semantics: from no semantics using tags/keywords such as images annotated on Flickr[9] , to more advanced knowledge representations, such as taxonomies or ontologies (e.g text annotations made by OpenCalais[10]). As a classic example, comparing a keyword-based approach and a semantic approach, an item referenced by specific keywords such as football, baseball, basketball, will not be recommended to a user interested in something more high level such as team sport. Through a semantic representation, the tacit knowledge that football, baseball and basketball are team sports is represented and can be used to extract pertinent user preferences and recommendations, and content about all type of sports teams would have been recommended because of the semantic link between the concepts. Metadata described with semantic concepts enable the use of more sophisticated recommender algorithms, such as [46] that takes into account the existence of "related" items according to their position in the ontology hierarchy; or spreading of semantic preferences (i.e. the extension of ontology-based user profiles through the semantic relations of the domain ontologies) as described in [47]. For collaborative filtering techniques, taking into account semantic similarities in the process has proven to increase accuracy and help reduce the sparsity problem [35]. However, there are some practical issues to be considered; for example, semantic reasoning induces some processing cost, that may not be feasible on the smallest devices. In addition, if the metadata are not sufficiently semantically richly described, some techniques enable the enrichment of the level of expressiveness of metadata, e.g. the matching of social tags with ontology concepts through Wikipedia[11][18].

Quantity: The amount of metadata is an important factor to consider: too few metadata may lead to inaccurate recommendations, whereas too much metadata may lead to useless processing. In addition, metadata description may vary in terms of depth (degree of precision) and breadth (variety of description). The risk with superficial description is to propose items to users that do not correspond exactly to what they expect. For example, if a user is interested in news related to natural disasters: with an in-depth description of the content, the user will have recommendations about different types of disasters; with a breadth-wise description of the content, he will have different points of view about the disaster (political, sociological, economical, etc.). Very in-depth descriptions may reinforce some drawbacks of

[9] Flickr, `http://www.flickr.com`

[10] OpenCalais, `http://opencalais.com`

[11] Wikipedia, `http://www.wikipedia.org`

content-based filtering algorithms, in particular in terms of overspecialisation. The level of depth or breadth the user wants may be learned by the system.

Description based on standard: Regardless of their level of expressiveness, the metadata can be described in different ways: using standards such as: Dublin Core[12], MPEG-7[13], IPTC[14]; or using proprietary formats. If the metadata representation is not imposed on the designer of the personalisation application, the choice of metadata representation may be guided by several considerations such as the degree of integration between the recommender and other parts of the application (e.g. use in a retrieval engine).

3.3.3 Understanding the properties of the item set

Volume of items: in addition to the quantity of metadata per item, we should consider the volume of items in the data set. It represents an important factor in determining the choice of a recommender system family. Indeed, collaborative filtering algorithms require large datasets and /or large user sets in order to compute correlations efficiently. This is typically appropriate for books, films, etc, whereas content-based algorithms can cope with a smaller size of data set (e.g. TV programmes).

Distribution of items: It is also essential to consider how items are distributed among the data set. For example, if in a movie data set, a very high proportion of items is annotated with the concept "action", this metadata may not be discriminative enough (too many content items selected if the user likes action films, too few if he does not). In such cases, and if possible, the level of depth of the annotation of data should be rethought in order to obtain better quality metadata.

Nature of items and stability of item set: News items, books, or TV programmes are intrinsically different, and therefore users do not behave the same way depending on the nature of the items. For example, it is relatively easy to conceive that interests related to TV programmes or books are more stable than the ones related to news, because news items change more frequently, and so there are more diverse subjects that can emerge. So, this criterion impacts the use or not of an adaptive recommender system (i.e. where the user profile evolves with time), and highlights the need to understand different evolution patterns, such as stable interests, progressive interests, fast changing interests, etc. [38, 39]. One should also consider the stability of the item set, i.e. how often new items are introduced or items disappear. In addition, a recommender can work with homogeneous data (e.g. movies only as done in MovieLens) or can manipulate heterogeneous content (as done in Amazon). In order to choose an appropriate strategy for item recommendations, it is important to study the correlation between the items, for example how an interest for a music type is reflected in book tastes. This analysis can help choose a strat-

[12] Dublin Core Metadata Initiative, `http://dublincore.org/`

[13] MPEG-7 overview, `http://www.chiariglione.org/mpeg/standards/mpeg-7`

[14] IPTC (International Press Telecommunications Council), `http://iptc.org`

egy for recommendations, such as considering all items globally, or instead apply specific rules based on the specificity of each item family.

User items ratings: Another important consideration is that of taking into account user ratings or not. If there is no user rating related to items in the system, this excludes the whole family of collaborative filtering methods. User ratings about items can be either *explicit* (for example on Amazon, users can indicate how much they enjoyed a book or a CD), and may be expressed on different scales (e.g. binary, multi-level). If this is not directly available, but thought to be useful in the personalised application, it is possible to compute *implicit feedback indicators* [41] (for example, the fact of purchasing an item can be considered as an implicit indicator of user interest), which can be used either in some recommendation algorithms (such as collaborative filtering), or as an input to a user profile adaptation module that will update users' interests based on likes and dislikes of specific items.

3.4 A method for using environment models

In Tables 1,2,3, we introduced three models to help understand the environment of the future recommender system. For each model and each feature we proposed some guidelines to define requirements and constraints on the future recommender system. To fully understand the importance of those features we propose a method in two steps:

1. *identify the dependencies between features*: the designer must find which features are influencing others by building a dependency graph across the three models. This graph will help the designer understand how a change on one feature impacts the overall recommender environment. For example, changing the integration of recommendations with navigation features (application model) will change user expectations (user model).
2. *identify key features of the models*: key features are the ones that have the most important impact on the choice of the recommendation and adaptation algorithms, and recommender architecture. For example, the performance correctness (application model) has a significant impact on the recommender choice. The identification of these key features helps to prepare the evaluation framework and understand how to interprete evaluation results.

As an illustration of this method, an example is given in section 5.

Thus, in this section, we have identified all the constraints that should be studied when designing a recommender system. Based on this first step, the designer should be able to determine the appropriate algorithms for filtering the items, and for adapting if necessary the user profile, and can choose the right architecture. The next phase for the designer consists of implementing his recommender system (see our references) and then in evaluating the algorithms, as explained in the next section.

4 Understanding the recommender validation steps in an iterative design process

Even though the models presented above allow making informed decisions about the crucial elements of a recommender system, many parameters have to be adjusted before it can be deployed, making lab testing and iterative design necessary. Two aspects must be studied: validation of the algorithms themselves and validation of the recommendations, in which users must be kept in the loop.

4.1 Validation of the algorithms

Many experimental papers have demonstrated the usefulness of testing a recommendation method against existing datasets such as MovieLens, Netflix[15], etc. From experience, such datasets are useful to understand the behaviour of an algorithm; they must sometimes be transformed and enriched in order to be usable (e.g. enrich MovieLens with IMDB[16] data). Experiments with datasets have been widely studied in the literature. For example, [25] gives a very exhaustive list of metrics that can be computed. In addition, they enable the support of objective comparison with other methods based on the characteristics and available information in the collections and may be particularly useful when large sets of users are needed. Such an evaluation also allows tweaking a number of personalisation parameters of the algorithm, e.g. distance metrics to compute the similarity between a user profile and a piece of content. These methods are widely explained in the literature and therefore are not further detailed here.

In this particular step of testing the algorithm on available dataset, the feature dependencies and impact graph as determined by the method described in section 3.4, must help the designer discover if the dataset is changing the recommender environment (data, user, application) compared to the final targeted system. This graph should help interpret the results of the algorithms and determine how far tweaking of the algorithms and testing should go. Indeed, if the environment properties are really close to the ones of the targeted final system, it may be worth adjusting the algorithms as much as possible. But if significant differences have been analysed between the experimental environment and the targeted environment, this phase of experiments with existing datasets should probably be kept as short as possible and should mainly focus on debugging rather than improving the accuracy of the recommenders by small amounts. Many research papers have focused on improving the accuracy of the algorithms [33], but even if a recommender algorithm provides good accuracy results, this is still with respect to the instance of the data and user models considered and associated with the dataset. Therefore, it is very important to understand the importance of the features of the environmental models and their

[15] Netflix dataset used for their competition, `http://www.netflixprize.com/download`

[16] IMDB, the Internet Movie Database, `http://www.imdb.com/`

interdependencies to interpret correctly the evaluation results against a standardized dataset. For example, when we move from TV data to news data, the behaviour of the user is not the same, and the data do not have the same dynamics (see section 5). In such a case, spending too much time tweaking the algorithm to improve the accuracy on the TV dataset is certainly a waste of time.

4.2 Validation of the recommendations

Whilst there are technical strategies for mathematically evaluating the accuracy of promoted items offered by a recommender, the acid test of ensuring measures of likely performance and satisfaction in the field are ultimately obtained through evaluating all areas of a recommender system *with end users*, with the focus upon making improvements to performance and increasing end user satisfaction.

The key to evaluating any interactive system is to follow a user-centred approach. This means evaluating *early*, *iteratively* and *frequently* based upon the changes and improvements made. This poses some challenges. Indeed as discussed earlier, recommenders are rarely at the centre of a development activity but instead are an integrated sub-feature of a much larger product or system. Common scenarios are that the development of the larger host system runs at a different development timescale to the recommendation engine. Such a situation poses problems for evaluation of the underlying recommender technology, leaving no method to collect user preferences upon which to build a user profile or a way to present recommendations to users.

These considerations often lead to the evaluation of the user-facing interaction aspects of the recommender (the data collection processes and the recommendation presentation) being split from the underlying engine, at least until both systems reach a prototype integration. From an evaluation perspective, this can sometimes make a lot of sense e.g. by preventing research findings from being confounded by competing negative or positive responses to other areas of the design.

There are many possible techniques in the user evaluation toolkit which can be deployed to investigate recommender developments. Some useful selective methods are introduced below together with when and in which context they can best be used. Each have been used in real life projects and are based upon our experiences of conducting user research in the area of recommender systems.

4.2.1 Card Sorting

Card sorting is not strictly an evaluation method but in fact a useful formative design research tool. It is used to create taxonomies of items based upon the naturalistic mental models users hold for the relationships between content or concepts. The method basically consists of asking users to sort a collection of cards, each which depicts a content item or sub classification, into groups based on similarity [43]. The resulting sorted groupings from a number of users can be analysed using cluster

analysis in order to find common shared patterns in user classification and to build a dendrogram of the associations. The time and complexity of this analysis can be reduced greatly by using one of the software tools such as IBM's EZsort [20].

We used this method when *investigating distance functions for television genre classifications.* Though many metadata structures exist in this area, it is very difficult without testing with users to understand if any one genre is perceived as being more similar to any single other genre more than others. As example, one result from the study showed that users classified a number of diverse genres as being close together due to a perception of them all as representing lighter viewing. This included genres such as game shows, comedy and soap operas. This was in contrast to more factual content genres classified as being less similar such as news and documentary. This information was integrated into the algorithm development to improve the accuracy of the recommendations.

The value of carrying out such an exercise is that the similarity measures for content items are based on a taxonomy created by real users, not artificial structures created by programmers or engineers. This method also unearths the subjective relative distances between disparate categories. These two factors increase the likelihood of recommendations being perceived as relevant and accurate from the perspective of the user. Due to its benefits in construction of the underlying algorithms, this tool is best used during formative development.

4.2.2 Low fidelity prototyping

Low fidelity prototyping is the umbrella term for a range of methodologies which cover exploration of an early interactive design idea with users. Traditionally this methodology has used paper prototyping to evaluate design ideas for how recommendations are to be presented, and also how user profile information may be captured and managed very early in the development of the recommender system. The major benefit of the method is that an evaluation can be achieved early without the need for costly and time consuming development. The method has no value however in evaluating the actual recommendation engine. It is executed in the form of a semi-structured interview using the prototype as a visual prompt. See [49] for a good introduction to the method and useful examples. When working with low fidelity prototypes it is important that they should not look like finished designs as this constrains users when discussing design ideas. Simple sketches and outlines allow users to openly postulate on highly valuable improvements and design ideas precisely because the designs appear so unfinished. This method is qualitative in nature as it attempts to gather rich data on the opinions of users. Evaluating a design with three or four users from each identified user group will often provide enough insight to significantly influence the design in a positive way.

We have used this type of investigation routinely to evaluate with users both actual design concepts but also general design principles. As an example, we used this methodology to introduce possible ideas for explicit rating scales in order to collect feedback which could be used to build user profiles. Simple line drawings

were enough to allow users to discern the differences between each concept and provide insights on their own perceived benefits and drawbacks for each. It allowed them to ponder more generally on the concept of providing explicit feedback to a recommender system, and what this meant in terms of both effort and privacy.

4.2.3 Subjective qualitative evaluation

This method is aimed at *evaluating the accuracy and satisfaction levels* attained by the recommendation engine from the perspective of end users. The major benefit of this method is *versatility*: it is a *qualitative* method which can be used as soon as a system is able to output recommendations even if the supporting data collection and recommendation presentation components do not yet exist. It can equally be used to evaluate fully implemented systems. The process is based on subjective assessment and a useful way of applying the method is through either a facilitator administered or participant completed standardised questionnaire[17]. Each recommendation generated for the user is presented within the questionnaire and a number of questions posed regarding the user's opinion towards it. The basic concept of the method consists of three steps.

1. Collection of users' data in order to build user profiles: the amount of data which needs to be collected is a function of the performance of the recommender. However it might equally be advisable to create participant samples within the study from whom varying amounts of information are captured to represent various typical usage patterns at given points in time, for example after two weeks. This allows the effect of the recommender's learning curve on user perceptions of accuracy and satisfaction to be analysed. The ease with which data can be collected from users is again a function of the maturity and design of the recommender. If the system is at a design stage where the preference data collection functions are operational then it may be possible to consider employing them within the study. This is especially true in systems where *implicit usage data* needs to be collected as it can be very difficult to elicit this type of data directly from users in other ways. Obviously, it is critical that the development team can access the actual user. Therefore the use of previously collected user data (for example in the form of web analytics) is unusable from the perspective of direct evaluation. Collecting implicit data from scratch requires the recruitment of participants for longer term monitored trials. This requires the use of techniques such as video observation or remote usage tracking in order to capture accurate preference data which can then be extracted and applied to implicit learning rules. This can be a laborious research activity but does also provide the added benefits of capturing real user insights and novel examples of use, which in turn drive innovation. In contrast, *explicit data collection* is far easier to simulate without a working system. Collecting feedback through an electronic or paper based survey form is a good method. An example we used was a simple spreadsheet. Users were asked to rate forty pieces of content on an explicit scale

[17] Questionnaire development is a science so if the techniques of questionnaire design and attitude assessment are new to you a good practical reference guide is [36].

which was being proposed for the finished system. Fifty screened participants were recruited remotely via the internet and the survey sent via Email. The users simply filled in their responses and returned them to the design team.

2. Generation of recommendations: The preference data returned by participants is extracted from the survey responses and coded for input into the recommender engine. An important consideration in providing individualised recommendations is to keep track of the user ownership of data and feedback responses throughout the study. Firstly, obtain permission from the users to retain their information for the duration of the study. Give each participant an ID, and use that number on every document throughout the evaluation process. Securely retain the participant details for the duration of the study against which the participant ID can be resolved. Finally ensure that participants' sensitive personal details are destroyed after the study and the results of the study anonymised. This simple process maintains user privacy (and data protection).

3. Presentation of recommendations: once recommendations for each individual user have been generated then they need to be given back to participants for evaluation. The questions to ask participants in the questionnaire really depend upon the goals of the recommender. Three common user perspectives on the recommendations that are likely to be of interest are:

- Does the recommendation match their preferences?
- Is the recommendation of interest?
- Would they be satisfied in a system that delivered such a recommendation?

Importantly questions should primarily be phrased around specific recommended items, not the recommendation list as a whole, because they are less likely to provide insights that can be used to improve the recommender. Receiving information on individual items allows weaknesses and bugs in the recommender design to be identified. For a project related to TV recommendations that we worked on, it allowed investigation as to why a successful recommender was receiving small numbers of extremely poor anomalous outlier satisfaction ratings. Investigation of the outliers allowed the discovery of foreign language content recommendations not identified by the metadata. The design team made a fix to more intelligently identify and handle foreign language content of this type and solved the dissatisfaction issue.

Analysis of the data can allow in-depth pictures of likely user satisfaction. It can also allow direct comparisons to be drawn between competing systems which can be used either for benchmarking purposes or fed back into design direction decisions.

4.2.4 Diary studies

Once a recommender is out in the field, how can information be gathered as to the success of the development over the medium to longer term? Collecting usage metrics does not capture the subjective motivations, pleasure or frustration in using a system. An important consideration for recommenders is that such systems can have *long learning curves* with a *changing user experience over extended periods of use*. In such cases analysis of the total experience of living with a system may

be more interesting than snapshots of satisfaction. Diary studies typically can run anything from a week to months, dependant upon the system under investigation and the opportunities to access engaged users. Diary-based study as a method is particularly well suited to mobile contexts, when direct observation or monitoring becomes logistically difficult. We have recently used this method for a study related to the evaluation of the impact of different usage contexts upon video content selection on heterogeneous mobile devices where no remote user logging was possible. The method proved very useful at identifying particular pain points for users.

Diary is in fact a very well establish method in many areas of social research, (see [10] for examples). The method relies upon a user to create a self-reported record of their day-to-day interactions with the system of interest (it relates to real user contexts described in their own words). The insight from such data can be integrated to improve systems already in use to uncover user requirements and contextual use issues for the next generation of development. However, in terms of logistics the method does have possible pitfalls to be considered:

- *recruitment and retention of users*: it can often be difficult to recruit participants for longer duration studies, and even more difficult to retain them;
- *non-reporting or false reporting of information in the diary*: completing diary entries takes effort and engagement which can be difficult for users to sustain over the whole duration of a study.

In order to run a diary study successfully it is very important to maintain contact with participants. Regular face to face or telephone debriefs encourage users to maintain their diaries by instilling the expectations of the researchers and allows the investigator to monitor the data collected and query incidents or comments closer to the times that events are reported. Such strategies can be used to identify critical incidents in use which have led to episodes of user satisfaction or dissatisfaction.

Whilst by no means an exhaustive list, this range of methods has significant utility in the development process from the early design stages, through prototype development and finally to release and post release.

5 Use case: a Semantic News Recommendation System

In the previous sections, we presented an approach to build a recommendation system, through the instantiation of three models: application, user, and data, which oblige one to think about possible choices for the recommender design. The purpose of this section is to compare what can be obtained from these models with what has been done in practice in the scope of a system - a News marketplace where news professionals or end-users can build, share (buy, sell) and consume multimedia content in a personalised way.

5.1 Context: MESH project

We illustrate the usefulness of the models that may help in driving the design a recommender system through the description of MESH[18] *(Multimedia sEmantic Syndication for enHanced news Services)*, a research project in which we participated that created a framework for intelligent creation and delivery of semantically-enhanced multimedia news information [51].

Fig. 2: Overview of the MESH platform

The main purpose of MESH was to provide news professionals and end-users with powerful but intuitive means to organize, locate, access and present huge and heterogeneous amounts of multimedia information. A MESH system is a news content broker that guides the user through a multimedia content web (the "news mesh"), finds automatically what he/she needs or wants, and presents it in the best possible arrangement on any terminal. Core ideas in the system consisted of (see Fig. 2): (1) Content delivered (push model) to users based on semi-automatically extracted semantic metadata and users' computed preferences; (2) Personalised multimedia summaries allowing easy digest of huge chunks of information, offering initial entry links to further access information through navigation aids that will prevent the "lost in multimedia cyberspace" syndrome; (3) Advanced syndication methods to allow rapid delivery of news from the end-sources (journalists) to the end-users (public) through fixed and mobile channels, proposing new flexible business and collaboration models.

In this context the role of the personalisation was to: (1) enable the system to proactively *push personalised news* items or personalised multimedia news summaries to user ; (2) provide support to users (both professional and end-users) to provide a *personalised search* of news items.

[18] MESH IST project (FP6-027685) (03/2006 - 02/2009), `http://www.mesh-ip.eu`

5.2 Environmental models in MESH

Based on the purpose of the application described above, we can illustrate how to instantiate the environmental models we described in section 3, in order to determine constraints on the recommender design.

5.2.1 Instantiation of the environmental models

Table 4: Application model instantiation

Recommender purpose	*increase system efficiency*, both for professional users (save time when gathering news information for example to build a "dossier" on a specific theme) and for end users (receive or browse relevant news with respect to their profile, recent interests, current situation or recent queries)
Recommender type	*multiple items*, provides a list of recommended multimedia news documents
Integration with navigation features	*tight*: e.g. through coupling with the information retrieval to deliver a personalised search service
Performance criteria	different criteria have a primordial importance depending of the kind of users of the system: journalists and professional users are much more interested in characteristics such as *correctness*, *response speed*, *reliability* and *robustness to attacks*, whereas end users are more interested in *correctness*, *transparency*, *serendipity*
Device to support the application	mainly *fixed* devices (only a limited subset of the application is available on a mobile terminal)
Number of users	*single user* application
Application infrastructure	*browser-based* application, with two modes of content delivery: pull mode (personalised search) and push mode (proactive delivery of multimedia news summaries)
Screen real-estate	*not limited*

Table 5: User model instantiation

Demographic information	only the *job* is considered as a discriminatory demographic factor: we distinguish between professional users such as journalists, photographers, etc. and end-users
Goal's existence and nature	in pull mode, explicit goals are expressed through user queries in the retrieval engine; in push mode : end users have no other goal than being informed about their favourite subjects and headlines news, Professionals still have the underlying goal of their current task that must be detected and inferred from their reading of items through the pull mode

Level of expectation	*medium*: the users can still use the system if the recommender does not work or perform poorly (though with a degraded efficiency); the recommender is not considered central to the user's activity
Change of expectation over time	*yes*: expectations of users should increase when they progressively discover the benefits of personalised search functions
Importance of user situation	*high*: for example, the users do not consume the same kind of news at home, at work or during holidays
Social environment	*alone*: by nature news consumption is rather an individual activity
Trust and privacy concerns	Not considered (assumption of a necessary trade-off between user benefits and divulgation of personal information and data)

Table 6: Data model instantiation

Data type	*semi-structured*: news items are described through a number of categories and concepts; methods to extract metadata from unstructured parts of the items (text, speech, video) are applied
Quality of metadata	overall, *medium*: good expressiveness through semantic annotation of content, but many metadata extracted by semi-automatic means (less reliable)
Description based on standards	yes: use of ontologies[19] based on IPTC
Volume / diversity of items	huge number of news items on a large variety of topics
Distribution of items	long-tail and mainstream: wide distribution of concepts, from those about mainstream events such as Obama's election to those related to the discovery of new species of frogs
Stability and persistence of items	the news item set is not really persistent, as it constitutes a continuous stream, therefore the items taken into account for recommendations are changing (in general news older than x days are ignored)
User ratings	*explicit* ratings (such as star rating) as well as *implicit* ratings (by monitoring clicks and time spent on news items) are considered

5.2.2 Links between the different models and constraints on design

In the MESH project, the study of the recommender system environment shows the important features of each model, how they influence each other, and which features have a direct impact on the choices about filtering/recommendation algorithms, infrastructure of the recommender and adaptation recommender option (see Fig. 3). This schema is essential in our method for analysing the key features of the system, and helps in determining that when a change occurs in the instantiation of one of the models, what the impact is on system constraints and requirements.

Impact on recommendation algorithms: In MESH, the primary purpose of the recommender is to improve efficiency of the system; therefore the selected algorithm should overcome limitations of content-based recommenders and of collaborative filtering [7, 14, 1]. The use of hybrid algorithms [14] can overcome these limitations. In addition, several categories of users cohabit, who may have different goals and interact with the system according to two different modes (push/pull). In pull mode, because the recommendations are driven by user queries, it may be preferable to use a content-based solution, whereas in push mode, for end-users, it may be interesting to have more diversity brought by collaborative filtering methods.

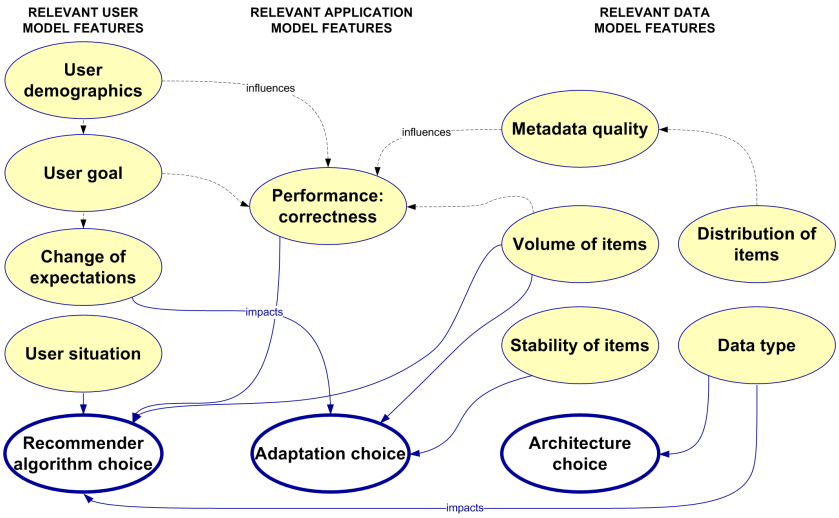

Fig. 3: Dependencies and impacts of model features in MESH

Therefore, it appears that hybrid solutions, combining content-based approaches and collaborative filtering may be more appropriate than one single approach. Another argument is related to the metadata: although there are metadata (which makes possible the use of content-based methods), their quality is still medium, which can lead to average quality recommendations, regardless of the intrinsic quality of the recommendation algorithms. Therefore it is safer not to rely solely on content-based methods.

Once it is decided that it may be worth combining different kinds of recommenders, the designer should analyze the environmental elements that help choose more precise methods within these families. Thus:

- for the collaborative filtering (CF) method: the study of the data shows that it is quite unlikely to apply a classical item-based CF [44], because due to the high volume and the dynamic nature of the data (news items arrive every few minutes), it is quite unlikely to have a lot of user ratings (either implicit or explicit) on specific news items. This particularly big sparsity requires the designer to imagine other ways of doing collaborative filtering, e.g. exploit user profile information when calculating user similarities in user-based CF [15].
- for the content-based algorithm: the highly structured metadata (expressed through ontological concepts) allow the use of more sophisticated recommenders that can exploit this structure to enhance the accuracy of recommendations [17].

Lastly, another important point is to determine how to combine these algorithms. Based on the environmental study, several heuristics are possible: static or dynamic combinations based on user type (end-user vs. professional) or based on the application mode (pull vs. personalised push). The user's context may also be used to parameterize the different algorithms and their combinations.

Additionally, in the context of news, it is essential to consider that there is a lot of similar content (such as news about Obama's election but coming from different sources); therefore some additional mechanisms to avoid recommending duplicated or equivalent content may be needed - for example clustering methods. In that case, it is important to identify and integrate into the overall recommendation process the criteria that users usually apply to distinguish between two similar news items.

Decision of building an adaptive recommender: the study of news data shows that news items change often (new news items are arriving fast) and also topics change often. Because of these properties (high variation and high coverage), user studies we have conducted have proven that users are not able to express exhaustively their interests relating to news, and that therefore a user profile may vary a lot. In addition, in the system, users have the opportunity to freely browse through specific news they are interested in. This information about possible new topics of interests for the users must be taken into account into their profile to maintain them up-to-date, so that it can be used also to deliver truly relevant personalised news in push mode. So, this makes it necessary to have an adaptive recommender, where user profiles evolve automatically and rapidly over time[38].

Impact on architecture: from the beginning, as MESH was a new application, we had two options for the recommender system: 1) a distributed solution or 2) a completely centralised solution. A distributed solution had some advantages: better management of privacy, and application independence, but some serious drawbacks: in the case of use of the application from different devices, it required profile synchronization between devices, data transfer between the devices and therefore back-end costs could be high. Therefore, it was decided to favour a centralized system, which eases the integration of the recommender with the other parts of the system such as content retrieval or content syndication.

5.3 In practice: iterative instantiations of models

As the recommender system was built in parallel to other modules of the system, it was not possible to wait until we had the real data to start building the recommender. So, it was decided to build algorithms and evaluate them with publicly available datasets:

- for the recommender algorithms: all tests were done with MovieLens + IMDB;
- for the profile adaptation algorithms, BARB[20] data. The choice of this dataset was driven by the need to have temporal evolution of content consumption.

The results showed that the hybrid recommendation methods developed in MESH provided more accurate results than state of the art methods (content-based or collaborative filtering). In this case, the quality of metadata was close to what we could expect from MESH and the user context (situation and goal) was not taken into account. However, in the experiments we carried out with the BARB dataset, which

[20] BARB - Broadcasters' Audience Research Board, `http://www.barb.co.uk`

collects data related to TV programme consumption during a 6 month period, we unconsciously created a new data model and a slightly different user model:

- Data model: items were TV programmes described by limited metadata, that were high level categories of programmes and a set of keywords extracted from the programme description text. The item distribution was mainstream with a poor to medium quality of annotation (there is significantly less diversity in TV programmes than in news). The stability of items was also different since TV programmes were predefined for a long period and we can consider them as quite stable (TV is fairly repetitive, and therefore new programmes do not appear every days). Two features influencing the performance of the recommender were changed by this dataset: *metadata quality* and *distribution of items* and one feature influencing the adaptation results was changed: *stability of items*.
- For the user model, professionals were not represented in the BARB dataset, so no specific goal was linked to the recommendation. It had an impact mainly on the correctness of the recommendation.

The BARB dataset was supposed to help us in evaluating our learning algorithm for user preferences but the results showed that after one week of learning, the user profiles remain stable, meaning that the preference learning mechanism had no further effect on the user preferences [38]. We had to interpret our results not only according to the behaviour of the algorithm by itself, but to the behaviour of the algorithm and the characteristics of the chosen dataset. The algorithm was supposed to deal with a large volume of items, not persistent and with mainstream to long-tail distributions with a good quality of metadata. It was supposed to be very reactive and built based on an average consumption of items that was largely superior to the TV programmes consumption per day. We learnt through this experience that changing the recommender environment (data, user and application models) has a huge implication on the expectation of the result and in the interpretation of the filtering algorithm and other mechanisms linked to the recommender system.

Therefore, we tried to use data as close as possible to the ones being used within MESH. We launched new experiments with a tool called "News@Hand"[16]. News@Hand combines textual features and collaborative information to make news suggestions to an end-user. News items are automatically and periodically retrieved from several on-line news services via RSS feeds. The title, summary and category of the retrieved news items are annotated with concepts from the system domain ontologies. During a period of 3 weeks we performed subjective qualitative evaluations (see section 4.2.3) to analyse the evolution of the preference learning algorithm with twenty users (they were divided into sub-groups in order to compare the evolution in different configurations of the algorithm). As the data model for this experiment was equivalent to the one in the MESH project and the user model just slightly different (no goal), we obtained results that were closer to our expectations (increase of recommendations quality over time).

By carefully studying the recommender environment through the instantiation of the data, user and application models and by identifying dependencies and impacts

of model's features, we think that critical issues must be addressed at design time in order to avoid errors and avoid losing time in dead ends.

6 Conclusion

In this chapter, we showed that though the technologies available to those who want to implement a recommender system are numerous, diverse and often hard to compare, a systematic evaluation of the environment is a key to making appropriate choices. We proposed for this to build three models describing the features of the environment that are the most influential to the recommender system design: application, user and data. Building these three models and studying their interaction allows narrowing the choice of appropriate recommendation algorithms, system architectures and user models.

In practice (as shown by the example we described), these three models by themselves will not allow to have a perfect design at the first go. But they are a very useful support for an iterative design methodology, which is the best way to go beyond technical excellence and reach the goal that matters in the end: user satisfaction.

References

1. Adomavicius, G., Tuzhilin, E.: 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 **17**(6), 734–749 (2005)
2. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, vol. 22, pp. 207–216. ACM (1993)
3. Ali, K., van Stam, W.: Tivo: making show recommendations using a distributed collaborative filtering architecture. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. Seattle, WA, USA (2004)
4. Anand, S.S., Mobasher, B.: Contextual recommendation. In: From Web to Social Web: Discovering and Deploying User and Content Profiles, Lecture Notes in Computer Science, pp. 142–160. Springer-Verlag (2007)
5. Anderson, C.: The long tail. Wired (2004)
6. Ardissono, L., Goy, A., Petrone, G., Segnan, M., Torasso, P.: Intrigue: Personalized recommendation of tourist attractions for desktop and handset devices. In: Applied Artificial Intelligence, pp. 687–714. Taylor and Francis (2003)
7. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Communications of the ACM **40**(3), 66–72 (1997)
8. Bernhaupt, R., Wilfinger, D., Weiss, A., Tscheligi, M.: An ethnographic study on recommendations in the living room: Implications for the design of itv recommender systems. In: EUROITV'08: Proceedings of the 6th European conference on Changing Television Environments, pp. 92–101. Springer-Verlag (2008)
9. Bias, R., Mayhew, D.: Cost-Justifying usability. Morgan Kaufman Publishing (1994)
10. Bolger, N., Davis, A., Rafaeli, E.: Diary methods: Capturing life as it is lived. In Annual Review of Psychology **54**(1), 579–616 (2003)

11. Bonnefoy, D., Bouzid, M., Lhuillier, N., Mercer, K.: More like this or not for me: Delivering personalised recommendations in multi-user environments. In: UM'07: Proceedings of the 11th international conference on User Modeling, pp. 87–96. Springer-Verlag (2007)
12. Bonnefoy, D., Drögehorn, O., Kernchen, R.: Enabling Technologies for Mobile Services: The Mobilife Book, chap. Multimodality and Personalisation. John Wiley & Sons Ltd (2007)
13. Bonnefoy, D., Picault, J., Bouzid, M.: Distributed user profile. Patent applications EP1934901, GB2430281 & WO2007037870 (2005)
14. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction **12**(4), 331–370 (2002)
15. Cantador, I.: Exploiting the conceptual space in hybrid recommender systems: a semantic-based approach. Ph.D. thesis, Universidad Autónoma de Madrid (UAM), Spain (2008)
16. Cantador, I., Bellogín, A., Castells, P.: News@hand: A semantic web approach to recommending news. In: Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2008), *Lecture Notes in Computer Science*, vol. 5149, pp. 279–283. Springer-Verlag (2008)
17. Cantador, I., Fernández, M., Vallet, D., Castells, P., Picault, J., Ribière, M.: Advances in Semantic Media Adaptation and Personalization, *Studies in Computational Intelligence*, vol. 93, chap. A Multi-Purpose Ontology-Based Approach for Personalised Content Filtering and Retrieval, pp. 25–51. Springer (2008)
18. Cantador, I., Szomszor, M., Alani, H., Fernández, M., Castells, P.: Enriching ontological user profiles with tagging history for multi-domain recommendations. In: Proceedings of the 1st International Workshop on Collective Semantics: Collective Intelligence and the Semantic Web (CISWeb 2008), pp. 5–19 (2008)
19. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: Gate: A framework and graphical development environment for robust nlp tools and applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02) (2002)
20. Dong, J., Martin, S., Waldo, P.: A conference on human factors in computing systemser input and analysis tool for information architecture. In: Conference on Human Factors in Computing Systems, pp. 23–24 (2001)
21. Duda, R.O., Hart, P., Stork, D.G.: Pattern Classification. John Wiley, New York (2001)
22. Gilb, T.: Principles of software engineering management. Arron Marcus Associates (1988)
23. Ha, S.H.: An intelligent system for personalized advertising on the internet. LNCS 3182 pp. 21–30 (2004)
24. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: CSCW'00: Proceedings of the 2000 ACM conference on Computer supported cooperative work, pp. 241–250. ACM, New York, NY, USA (2000)
25. Herlocker, J.L., Terveen, L.G., Konstan, J.A., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on information systems **22**, 5–53 (2004)
26. International Organisation for Standardisation (ISO): ISO 13407: Human centred design processes for interactive systems.
27. Jameson, A., Baldes, S., Kleinbauer, T.: Two methods for enhancing mutual awareness in a group recommender system. In: AVI'04: Proceedings of the working conference on Advanced visual interfaces, pp. 447–449. ACM, New York, NY, USA (2004)
28. Lhuillier, N., Bonnefoy, D., Bouzid, M., Millerat, J., Picault, J., Ribière, M.: A recommendation system and method of operation therefor. Patent application WO2008073595 (2006)
29. Lhuillier, N., Bouzid, M., Gadanho, S.: Context-sensitive user preference prediction. Patent application GB2442024 (2006)
30. Lhuillier, N., Bouzid, M., Mercer, K., Picault, J.: System for content item recommendation. Patent application GB2438645 (2006)
31. Masthoff, J.: Group modeling: Selecting a sequence of television items to suit a group of viewers. User Modeling and User-Adapted Interaction **14**(1), 37–85 (2004)
32. McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B., Nixon, P.: Cats: A synchronous approach to collaborative group recommendation. In: G. Sutcliffe, R. Goebel (eds.) Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference, pp. 86–91. AAAI Press (2006)

33. McNee, S.M., Riedl, J., Konstan, J.A.: Accurate is not always good: How accuracy metrics have hurt recommender systems. In: CHI'06 extended abstracts on Human factors in computing systems, pp. 1097–1101. ACM, New York, NY, USA (2006)
34. Mobasher, B., Burke, R., Bhaumik, R., Williams, C.: Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. ACM Trans. Internet Technol. **7**(4) (2007)
35. Mobasher, B., Jin, X., Zhou, Y.: Semantically enhanced collaborative filtering on the web. Web Mining: FromWeb to SemanticWeb pp. 57–76 (2004)
36. Oppenheim, A.: Questionnaire Design, Interviewing and Attitude Measurement. Continuum International Publishing, New York (2001)
37. Papadogiorgaki, M., Papastathis, V., Nidelkou, E., Kompatsiaris, Y., Waddington, S., Bratu, B., Ribière, M.: Distributed user modeling for personalized news delivery in mobile devices. In: 2nd International Workshop on Semantic Media Adaptation and Personalization (2007)
38. Picault, J., Ribière, M.: An empirical user profile adaptation mechanism that reacts to shifts of interests. Submitted to the 18th European Conference on Artificial Intelligence (2008). http://www.mesh-ip.eu/upload/ecai2008.pdf
39. Picault, J., Ribière, M.: Method of adapting a user profile including user preferences and communication device. European Patent EP08290033 (2008)
40. Ribière, M., Picault, J.: Progressive display of user interests. Tech. rep., Motorola (2008). https://priorart.ip.com/download/IPCOM000167391D/
41. Ribière, M., Picault, J.: Method and apparatus for modifying a user preference profile. Patent application WO2009064585 (2009)
42. Rich, E.M.: User modeling via stereotypes. Cognitive Science **3**, 329–354 (1979)
43. Rugg, G., McGeorge, P.: The sorting techniques: a tutorial paper on card sorts, picture sorts and item sorts. Expert Systems, The journal of Knowledge Engineering **14**(2), 80–93 (2002)
44. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW'01: Proceedings of the 10th international conference on World Wide Web, pp. 285–295. ACM, New York, NY, USA (2001)
45. Shepherd, M.: Tutorial on personalization and filtering on the web. Web Information Filtering Lab, Dalhousie University, Canada. http://ncsi-net.ncsi.iisc.ernet.in/gsdl/collect/icco/index/assoc/HASH011b.dir/Tutorial-Shepherd.ppt
46. Shoval, P., Maidel, V., Shapira, B.: An ontology-content-based filtering method. International Journal of Information Theories and Applications (15), 303–318 (2008)
47. Sieg, A., Mobasher, B., Burke, R.: Ontological user profiles for personalized web search. In: Proceedings of AAAI 2007 Workshop on Intelligent Techniques for Web Personalization, pp. 84–91. Vancouver, BC, Canada (2007)
48. Signa, R.: Design strategies for recommender systems. UIE Web App Summit. http://www.slideshare.net/rashmi/design-of-recommender-systems
49. Snyder, C.: Paper prototyping: The fast and east way to design and refine user interfaces. Morgan Kaufmann Publishing, San Francisco (2003)
50. Terveen, L., Hill, W.: Human-Computer Interaction in the New Millennium, chap. Beyond Recommender Systems: Helping People Help Each Other, pp. 487–509
51. Villegas, P., Sarris, N., Picault, J., Kompatsiaris, I.: Creating a mesh of multimedia news feeds. In: European Workshop on the Integration of Knowledge, Semantics and Digital Media Technologies (EWIMT), pp. 453–454. IEE (2005)
52. Yu, Z., Zhou, X., Hao, Y., Gu, J.: Tv program recommendation for multiple viewers based on user profile merging. User Modeling and User-Adapted Interaction **16**(1), 63–82 (2006)
53. Zhang, M., Hurley, N.: Avoiding monotony: improving the diversity of recommendation lists. In: RecSys'08: Proceedings of the 2008 ACM conference on Recommender systems, pp. 123–130. ACM, New York, NY, USA (2008)
54. Ziegler, C.N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: WWW'05: Proceedings of the 14th international conference on World Wide Web, pp. 22–32. ACM, New York (2005)