User Modeling and Personalization
8: Evaluation of Adaptive Systems

Eelco Herder

L3S Research Center / Leibniz University of Hanover
Hannover, Germany

6 June 2016

Outline I

Outline II

Correlations

Evaluation of Adaptive Systems

In this course, we focus on methods and techniques for user modeling and personalization, or - more specifically - on adaptive hypermedia systems.

Adaptive Hypermedia Systems

"By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user."

There are two main categories of adaptive techniques:

Adaptive presentation techniques work on the content level. Items that may be adapted include text, layout, graphics or any other form of multimedia. A significant amount of research has been carried out on *text adaptation*. The general goal of text adaptation is to hide some parts of information that are deemed not to be relevant for the user

Adaptive navigation techniques work on the link level. Disabling, removing or annotating associative links can help users to find relevant items more easily. Common forms of adaptive navigation techniques include conditional links, adaptive menus, breadcrumbs, graphical overviews, direct guidance

Particular attention has been paid to recommender systems:

Recommender Systems

Recommender systems work from a specific type of information filtering system technique that attempts to recommend items (movies, TV program/show/episode, video on demand, music, books, news, images, web pages, scientific literature such as research papers etc.) that are likely to be of interest to the user.

In this lecture, we focus on methods and measures for the evaluation of adaptive hypermedia systems.

Why testing and evaluation is important

"Every program does something right, it just may not be the thing that we want it to do"

The development of software systems involves a series of production activities where opportunities for injection of human fallibilities are enormous:

- errors may begin to occur at the very beginning of the process, where the objectives may be erroneously or imperfectly specified
- as well as in later design and development stages

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding.

Evaluations may serve different goals:

- *summative* evaluation aims to determine the value or impact of a system
- *formative* evaluation aims to identify shortcomings or errors in a system in order to further improve it and to guide the system design and development

It goes without saying that high-quality software is an important goal. But software quality is a complex mix of factors that varies across different applications and user contexts.
Hewlett-Packard uses the following main categories of software quality factors, abbreviated as FURPS:

- **Functionality** is assessed by evaluating the features and capabilities of the program
- **Usability** is assessed by considering human factors, aesthetics, consistency and documentation
- **Reliability** is evaluated by measuring the frequency and severity of failure and the accuracy of output results
- **Performance** is measured by processing speed, response time and resource consumption
- **Supportability** includes maintainability, testability, compatibility and configurability

Within (HCI) research and academia, researchers employ (black-box) testing and evaluation to validate novel design ideas and systems

- usually by showing that human performance or work practices are somehow improved when compared to some baseline set of metrics (e.g., other competing ideas)
- or that people can achieve a stated goal when using this system (e.g., performance measures, task completions)
- or that their processes and outcomes improve.

What makes evaluation of adaptive systems different

Adaptive systems are not directly user-controlled
The very aim of adaptivity is to imbue a system with intelligence that allows it to actively take the initiative in supporting the users' activities. Traditional evaluation approaches fail to address this.

The adaptation process often takes time
The system needs to learn about the user's goals, knowledge, preferences, etc., before adaptation can take place. The observation of any effects of adaptivity may require long-term studies.

The effects of adaptation depend on user and context

Different algorithms and approaches may be better or worse for different users. Many adaptive systems have been designed for a particular purpose, in a particular domain, with a particular kind of users. The algorithms and approaches used may perfectly make sense in this particular case, but may be entirely inappropriate in a different domain

- Recommending movies is quite different from recommending learning resources

Risks of Adaptivity

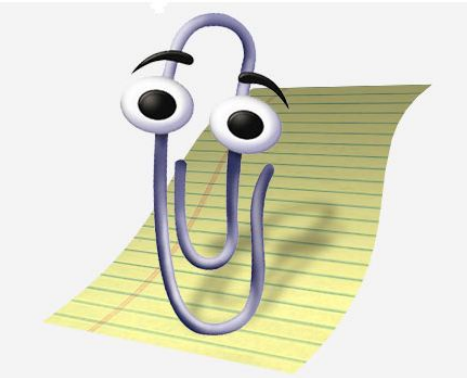
Adaptation is sometimes characterized as *deferred design*: instead of searching for a general design that is suitable for most users, adaptive interface designers build in mechanisms able to determine a user's particular needs and automatically adapt the interface accordingly.

Despite the potential advantages in terms of the system's general applicability and life-span, there are risks involved in not knowing exactly how the interface will behave.

Example: Microsoft Office Assistant

As an example, the Lumiere Project designed a Bayesian Network for inferring user goals and needs in Microsoft Office applications in order to provide users with suggestions relevant to their tasks.

The commercial Microsoft Office Assistant that was designed based on the results of the Lumiere Project used slightly less advanced reasoning mechanisms, which made the assistant's behavior less intelligent, but at least more predictable.



Layered Evaluation

A common approach to the evaluation of adaptive system is to compare it with a (non-adaptive) baseline system.

- "Is this (adaptive) version better than that (non-adaptive) version, in this particular respect"

A successful evaluation would prove the value and impact of the system, but:

- are the findings *generalizable* beyond this particular system (i.e. how generic is the design idea) ?
- why, and under what conditions, can a particular type of adaptation be employed?

Moreover, what if the evaluation results are not satisfactory?

- unsuccessful adaptations might be due to incorrect assessment results
- or to improper adaptations based on correct assessments
- or both ...

Overview

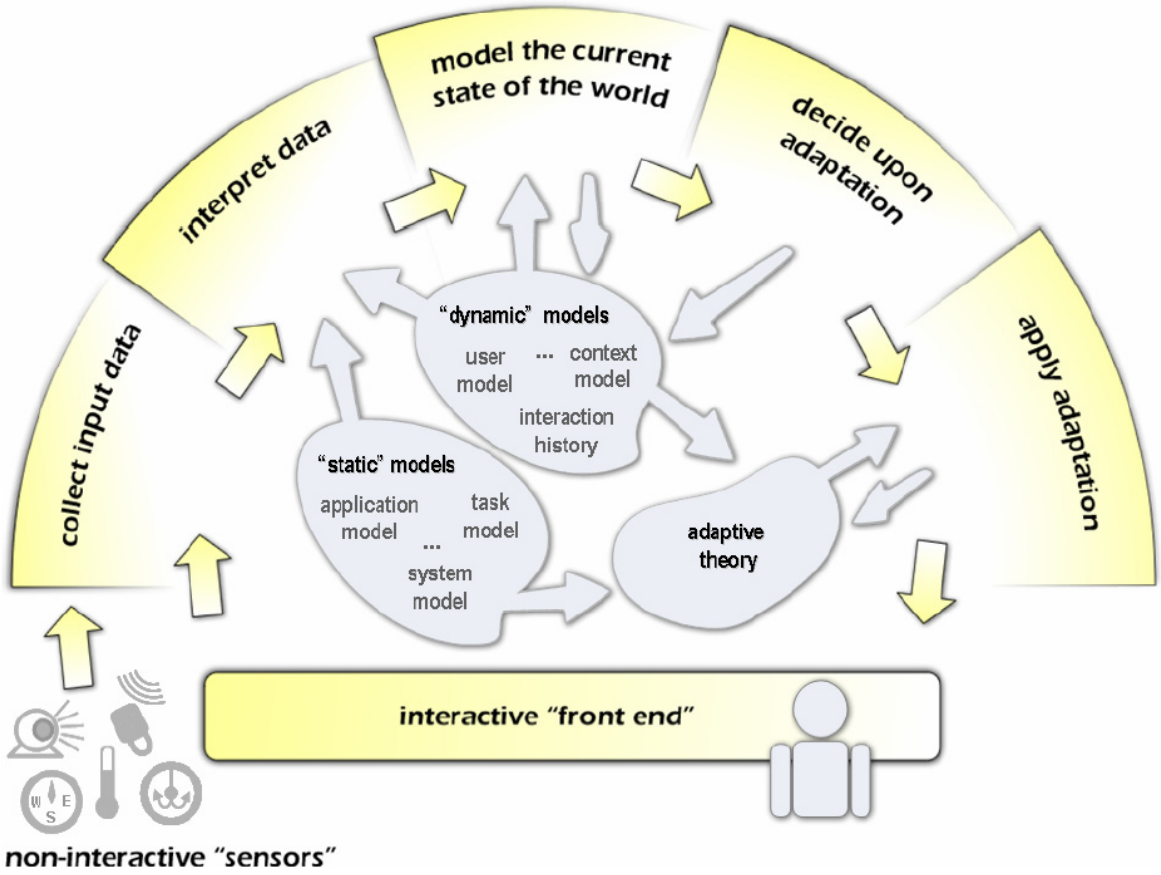


Figure: Composition of the adaptation process in layers (Paramythis, Weibelzahl and Masthoff, 2011)

Layered evaluation is a "piece-wise" evaluation of adaptation that provides insight into the individual adaptation stages:

Collection of input data refers to the assembly of user interaction data, along with any other data (available, e.g., through non-interactive sensors) relating to the interaction context.

Interpretation of the collected data is the step in which the raw input data previously collected acquire meaning for the system.

Modeling of the current state of the world refers to derivation of new knowledge about the user, the interaction context, etc., as well as the subsequent introduction of that knowledge in the dynamic models of the adaptive system.

Deciding upon adaptation is the step in which the adaptive system decides upon the necessity of, as well as the required type of, adaptations, given a particular state of the world, as expressed in the various models maintained by the system.

Applying (or instantiating) adaptation refers to the actual introduction of adaptations in the user-system interaction, on the basis of the related decisions.

The "big picture" is provided by the evaluation of the *system as a whole*

- by comparing it with a baseline system
- or checking that the stated goals are achieved
- or by showing an improvement in processes and outcomes

Step by step

1. Collection of input data

In this phase, the reliability and external validity of the input data is evaluated.

User data

User data consists of events and observations on the user's interaction with the system that can either directly be used for adaptation of that need to be resolved to user characteristics.

User data may be directly provided by the user, automatically logged by the system, registered via sensors, provided by another system, ...

But how **accurate** is this data?

- did users provide all their interests in their profiles
- does the server log contain all page requests (no missing entries due to caching)
- how reliable is location tracking via your mobile phone (and what about latency or sampling rate)
- how did Facebook deduce the user's interests?

Undetected problems in this layer may cause unexpected effects in other layers.

For example, inaccurate location tracking might lead to inappropriate travel directions or restaurant suggestions. When the level of inaccuracy is known, these kinds of problems can be avoided.

2. Interpretation of the collected data

Inference of knowledge

Knowledge inference is the process of interpreting events and observations on a user U, making use of conditions, rules or other forms of reasoning, and the storage of the inferred knowledge in the user model.

Many interactions contain meaning in themselves, such as page visits, bookmarking or saving actions, queries issued by the user and items inspected or bought from an e-commerce Web site.

Other interactions need to be combined or interpreted in order to become meaningful, such as key strokes, mouse clicks and eye gaze behavior.

For the interpretation of the data, systems often make inferences based on certain assumptions. These inferences introduce *uncertainty* and it is therefore a good idea to question the *validity* of the interpretations:

- is it true that if a user *visited* a page, that he *learned* the content of this page
- does a five-minute page visit in the server log indicate high interest or just a coffee break
- if a user indicates that he is not interested in a news story, could this also be caused by the fact that the user is in a hurry or that the user has read the story before

A measure to evaluate the validity of the interpretation, is by assessing how predictable the results are, using cross-validation:

Cross-validation

Cross-validation is a technique for assessing how the results of the inference process (or some other form of analysis) will generalize to an independent data set (other users, other user characteristics).

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set).

To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

Another measure is the *scrutability* of the user model: are users able to determine (inspect and control) how (or even whether) specific actions of theirs are interpreted by the system.

Scrutable user models allow users to determine themselves what is modeled and how adaptations based on their models will be conducted.

3. Modeling of the current state of the "world"

An intermediate point between the interaction assessment and the adaptation decision phase is the *user model*. The model may represent assumptions on user characteristics, or predictions on user actions, or interests that are closely related to adaptation decisions.

User Model

A User Model is a data structure that characterizes a user U at a certain moment in time.

The quality of the user model is a direct result of the validity, predictability and scrutability of the interpretation of the observed data.

Further evaluation criteria include:

Comprehensiveness:

- is the structure of the model expressive enough (e.g. does a flat model suffice or would a domain overlay be more effective)
- is the data format used expressive (*precise*) enough (e.g. does a boolean for 'knowledge' suffice)
- does the model support the intended adaptation or recommendation methods (e.g. can it be used for association rules)

Conciseness:

- does the user model contain elements that cannot be inferred from the user data (e.g. it may be hard to derive learning progress from keystrokes)
- does the adaptation rules assume that these elements are not empty

Sensitivity:

- how much data is needed for a 'complete enough' model for adaptation (cold-start problem)
- how quickly does the model change based on new data (too dynamic models may lead to 'chasing' effects, in which the user reacts to changes in the system and the system reacts to changes in user behavior and so on)

4. Deciding upon adaptation

Even though the adaptation decision (what will be recommended or adapted) and the application of the adaptation decision (the next phase) are often combined in a system, it makes sense to evaluate them separately:

- the adaptation decision involves the choice to provide recommendations of particular items, to provide the user with 'helpful' suggestions while writing a letter in Word, . . .
- the application of the adaptation decision involves how the adaptation is presented to the user (e.g. one random recommendation at a time, a weekly newsletter, a subtle message at the bottom of the screen or an embodied Office Assistant)

The primary aim of this step is to determine whether the adaptation decisions made are the optimal ones, given that the user's properties have been inferred correctly:

- is the decided adaptation necessary and/or appreciated by the user (does it make sense to recommend news items at all?)
- is the chosen adaptation method or algorithm the most effective one (e.g. is collaborative filtering the best recommendation method for news items - see second part of this lecture)
- will the method be accepted by the user ('My TiVo thinks I'm Gay')

Evaluation of recommender systems typically involves the usage of a standard dataset. The user interface is fixed as well.

5. Applying adaptation decisions

This stage directly addresses the way in which adaptations or recommendations are presented to the user. Obviously, *usability* is an important issue:

- how obtrusive or obstructive is the application of an adaptation
- can the user disallow, retract, or even disregard an adaptation
- does the user accept the adaptation

These issues are typically addressed with user studies - which will be the topic of the next lecture.

Evaluating the system as a whole

Testing and evaluation addresses quality factors such as functionality, usability, reliability, performance and supportability.

In research, evaluation mainly targets functionality (does the system perform as good as anticipated) and usability (do users like it).

In many cases, objective measures are rather straightforward, such as:

- for elearning: decrease of learning time, increased retention time of learned material
- for online stores: number of products purchased (and not returned)
- for contextual help: number of suggestions followed, reduction of error rates

In other cases, general measures or theory-assessment measures (improvement with respect to theories on user behavior) can be used, such as:

- user satisfaction (measured using a standardized questionnaire)
- interaction time with the system
- reduction of *behavioral complexity* in Web navigation

A typical evaluation attempts to show that:

- the system performs better than a comparable (non-adaptive) system
- the system achieves the stated goals
- the system performs better than some predefined thresholds

In all these cases, care should be taken that improvements are measurable and that any comparison with other systems is fair.

Even though of secondary importance, reliability and performance are known to have an impact on measures for functionality and usability.
Supportability is typically ignored in research-oriented evaluations.

Evaluating recommender systems

Evaluation of recommender systems usually focuses on the performance of a particular algorithm (collaborative filtering, content-based recommendation, hybrid recommenders, association rules, ...).

The best setup would be a live user experiment:

- a controlled study with participants assigned to one of the possible conditions (e.g. two different versions of recommenders)
- a field study where a particular system is made available to a community of users

Offline evaluation

As a quicker and more economical alternative, much of the work in algorithm evaluation has focused on off-line analysis of predictive accuracy.

In such an evaluation, the algorithm is used to predict certain withheld values from a dataset, and the results are analyzed using one or more *performance measures*.

This is actually Stage 4 of the Layered Evaluation approach.

Datasets

In the ideal case, a *natural dataset* is available, based on user logs from an earlier version of the system or a comparable system.

This may not be as straightforward as it seems:

- A movie recommender algorithm may be evaluated with the MovieLens dataset, which contains the available items and an extensive log of user ratings of these items
- The evaluation task would be to predict the actual ratings provided by the users, based on past interactions or commonalities with other users
- However, if the goal of your recommender system is to get users to watch or buy items that they normally would not consider, this evaluation task is slightly inappropriate

- This may be compensated by adjusting user ratings with (artificial) ratings based on e.g. content similarity
- A video rental store may use the MovieLens dataset as well, but the results may not be completely representative, because
  - the video store has other (less) items available than present in the MovieLens system
  - the decision to rent a video may be inspired by other motivations than the rating of a movie (e.g. price, special offers)

Synthesized datasets are datasets that are assembled by:

- combining user behavior from two different systems
- simulating user behavior using some algorithm
- having experts rating items on different scales
- ...

Synthesized data sets are often used for finding obvious flaws in recommender algorithms, as early steps while designing a complete system.

But it is risky to draw comparative conclusions from synthetic dataset, because

- the data may fit one of the algorithms better than the others (by chance or on purpose)
- it is not guaranteed that the dataset accurately models the nature of real users and real data

Bias

For both natural and synthesized datasets, it is important to evaluate possible biases introduced by:

- the way the data is collected or generated
- distribution of items and ratings in the original domain (e.g. movies in the MovieLens dataset)
- the way ratings were provided (scale of the ratings, motivation for rating, explicit versus implicit ratings)
- .....

Performance - Accuracy Metrics

Accuracy metrics

An accuracy metric measures how close a recommender system's predicted ranking of items for a user differs from the user's true (or estimated) ranking of preference. Accuracy metrics may also measure how well a system can predict an exact rating value for a specific item.

Criteria for accuracy (or goodness) vary from system to system and from situation to situation:

- If Amazon gives you five product suggestions, most of these suggestions should interest you
- If Amazon shows you a selection of products based on your query, this selection should contain most of their relevant offerings

Precision and Recall

Precision and recall are the most popular metrics for evaluating information retrieval systems (such as search engines).

They are commonly used for recommender systems as well.

Let D_{sel} be the set of items selected by the algorithm and D_{rel} the set of items that are relevant for the recommendation task.

Precision is defined as the ratio between the number of selected items that are relevant and the total number of selected items.

$$Precision = \frac{|D_{rel} \cap D_{sel}|}{|D_{sel}|}$$

Conversely, **Recall** is defined as the ratio between the number of selected, relevant items and the total number of relevant items.

$$Recall = \frac{|D_{rel} \cap D_{sel}|}{|D_{rel}|}$$

- A perfect precision score of 1.0 means that every selected item was relevant (but not whether all relevant items were selected).
- A perfect recall score of 1.0 means that all relevant items have been selected (but not how many irrelevant items were also selected)

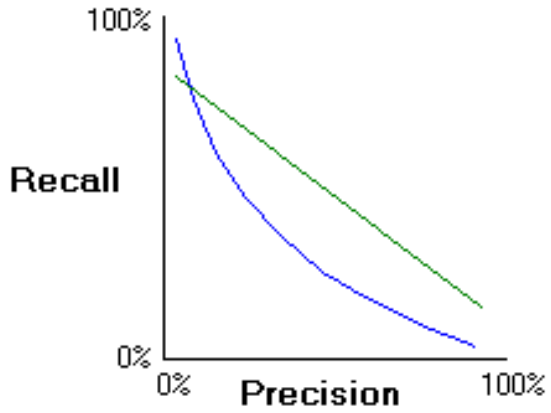


Figure: Two precision-recall curves. Note the blue curve, where the recall drops dramatically when precision gets higher - the result set gets better, but many relevant items are left out.

F-measure

Precision and recall are more or less inversely related. To determine the optimal number of items to be selected, one can compare the ratio between precision and recall for different sizes of items sets. This harmonic mean of precisian and recall is called the **F-measure**.

$$F = 2 * \frac{Precision * Recall}{Precision + Recall}$$

For recommender systems it is often hard to determine for each item whether it is relevant or not. Therefore, evaluation results often report precision for the top-k results: **P@k**.

Precision and Success at rank k (P@K and S@k)

The precision measure P@k assumes that the recommendation task is to generate a set that contains as many relevant recommendations as possible. It also assumes that the user will evaluate all these recommendations (e.g. by clicking on a list of search results).

S@k is an alternative for the P@k measure and stands for the mean probability that at least one *relevant* item occurs within the top-k ranked items.

S@k is a useful measure for situations in which the user is looking for a specific item or is happy with just one useful suggestion, such as:

- the user wants to rent just one video
- the user is interested in a specific news item (and is not interested in 10 other articles about the same topic)

Mean Reciprocal Rank

Used for evaluating any process that produces a list of possible responses to a query, ordered by probability of correctness. This measure is useful if a recommender system is used for generating or optimizing search results.

The reciprocal rank of a query response is the inverse of the rank of the first relevant item. The mean reciprocal rank is the average of the reciprocal ranks of results for a set of queries N

$$MRR = \frac{1}{N} \sum_{n \in N} \frac{1}{rank_n}$$

Correlations

Correlations between predicted ratings or rankings and the actual ratings or rankings indicate how well recommendations fit the user's actual preferences.
The most common correlation measure for ratings is the Pearson correlation measure:

$$r(pred, act) = \frac{\sum_1^n (pred - \overline{pred})(act - \overline{act})}{n * \sigma(pred)\sigma(act)}$$

The standard deviation σ is given by

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

Spearman's Rank Correlation

For correlations between the predicted and actual ranking, the **Spearman's rank correlation coefficient** ρ is used, which is defined similar in the same manner as the Pearson correlation. The only difference is that the predicted ratings are transformed into ranks and the correlations are computed on the ranks.

$$\rho(pred, act) = \frac{\sum_1^n (pred - \overline{pred})(act - \overline{act})}{n * \sigma(pred)\sigma(act)}$$

In statistics, Spearman correlations are used if the data does not follow a normal distribution or is otherwise skewed.

Kendall's Tau

Another ranking coefficient is **Kendall's tau** τ, which can be used if you have a small data set with a large number of tied ranks.

Kendall's tau is less popular than the Spearman's coefficient, but generally seen as more accurate.

Let N be the number of predicted rankings. Let C be the number of *concordant pairs*, pairs of any two of the N items that the system predicts in the proper ranked order.

And let D be the number of *discordant pairs*, pairs of items that the system predicts in the wrong order.

Kendal's tau is defined as the difference between the concordant and discordant pairs, divided by all possible item pairs.

$$\tau = \frac{C - D}{\frac{1}{2}N(N-1)}$$