

AWESOME - A Data Warehouse-based System for Adaptive Website Recommendations

ANDREAS THOR ERHARD RAHM

University of Leipzig, Germany

Presented by: Linas Baltrunas, Ivan Zorzi

Outline

- 1 Introduction
- 2 Architecture
- 3 Recommenders
- 4 Recommender Evaluation
- 5 Adaptive Recommender Selection
 - Query-based Top Recommender
 - Machine-learning Approach
- 6 Conclusions

Introduction

- How should recommendations be generated automatically to optimally serve the users of a website?
- The relative utility of recommendation approaches depends on the website, its users and other factors
 - ▶ There cannot be a single best approach
- Advanced websites support many recommenders but they are unable to select the most effective approach
 - ▶ Huge web pages with reduced usability

Introduction

Problem

- Choice of different types of information to generate recommendations
- Selection of effective approach per user or product
- Buying behaviour VS. usage (navigation) behaviour

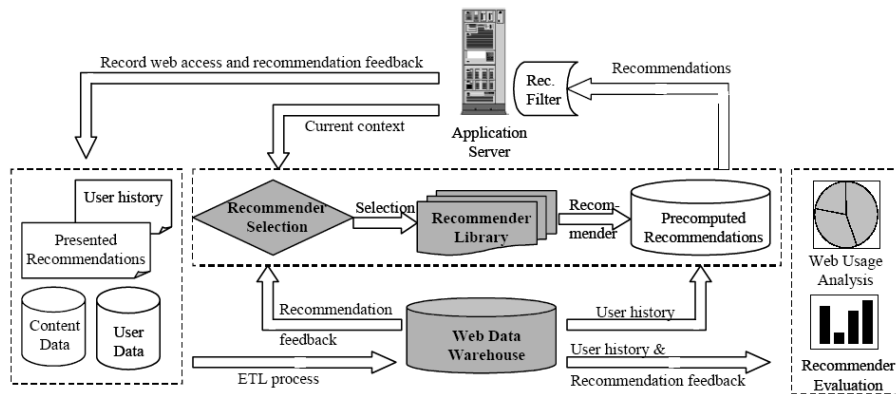
Introduction

Goal

- Use an approach for comparative quantitative evaluations of different recommenders
- Exploit web usage behaviour (feedback) to automatically and adaptively improve recommendation quality

- AWESOME (Adaptive website recomendations)
 - ▶ data warehouse-based website evaluation and recommendation system
 - ▶ extensible library of recommender algorithms that can be comparatively evaluated for websites based on user feedback
 - ▶ selection of recommender depends on the recommendation quality

Architecture



Architecture

- Application server running the website
- Each website access has a specific *context*
 - ▶ Recommendations are created dynamically depending on this context
- Recommendations are determined by a variety of algorithms from an extensible *recommender library*
- Recommenders exploit information on the usage history + additional information maintained in a *web data warehouse*

Architecture

Procedure

- For a given context AWESOME:
 - ▶ selects the most appropriate recommender(s) using some *selection rules*
 - ▶ selects the best recommendations for the chosen recommender
- Clear separation between the selection of recommenders and recommendations

Architecture

ETL process

- ETL (extract, transform, load) process is needed to refresh the data warehouse
- Web log files of the application server and other data sources are processed
- A specific application server logging has been adopted:
 - ▶ Web usage log file: advanced Common Log Format (CLF) of common web servers
 - ▶ Recommendation log file: records all presented recommendations
- The ETL process updates all affected warehouse dimension and fact tables

Architecture

ETL process

Page	requested page
Date, Time	date and time of the request
Client	IP address of the user's computer
Referrer	referring URL
Session ID	session identifier
User ID	user identifier

a) Web usage log

Pageview ID	page view where recommendation has been presented
Recommendation	recommended content
Position	position of this recommendation inside a recommendation list
Recommender	recommender that generated the recommendation
Strategy	strategy that selected the applied recommender

b) Recommendation log

Architecture

Eliminating useless information

- Several approaches are used to detect and eliminate web crawler requests
 - ▶ IP address list of known crawlers to avoid logging their accesses
- Sessions containing page views reached by following invisible links are eliminated

Architecture

Data warehouse

- The data warehouse is a relational database with a galaxy schema consisting of several fact tables sharing some dimensions
- Different fact tables are used for page views, sessions, purchases
- A recommendation fact table is used to represent positive and negative user feedback on recommendations
 - ▶ Each record refers to one presented recommendation
 - ▶ Three boolean measures are used to derive recommendation quality metrics
 - ★ *Accepted*: clicked or not by the user
 - ★ *Viewed*: recommendation was a useful hint or not
 - ★ *Purchased*: recommended product was purchased or not

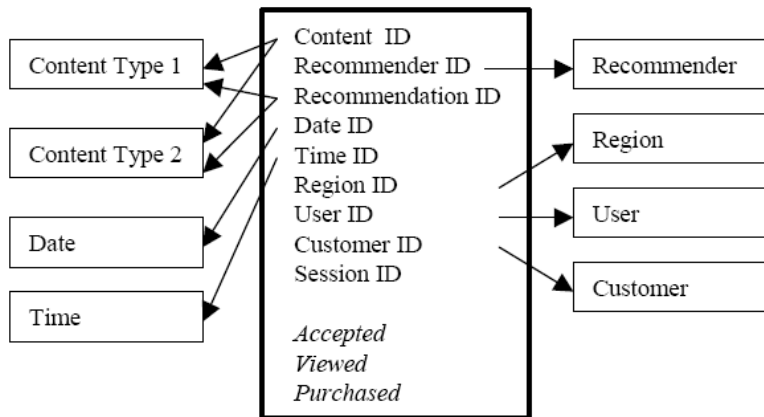
Architecture

Recommendation fact table

Dimension tables

Fact table Recommendation

Dimension tables



Recommenders

Intro

- Given a web page request specified by a context, a recommender generates an ordered list of recommendations
- Recommendations are usually presented as titled links with a short preview
- There are three types of context information relevant for determining recommendations:
 - ▶ Current content: currently viewed content and its related information
 - ▶ Current user: identified by a cookie
 - ▶ Additional information available from the HTTP request

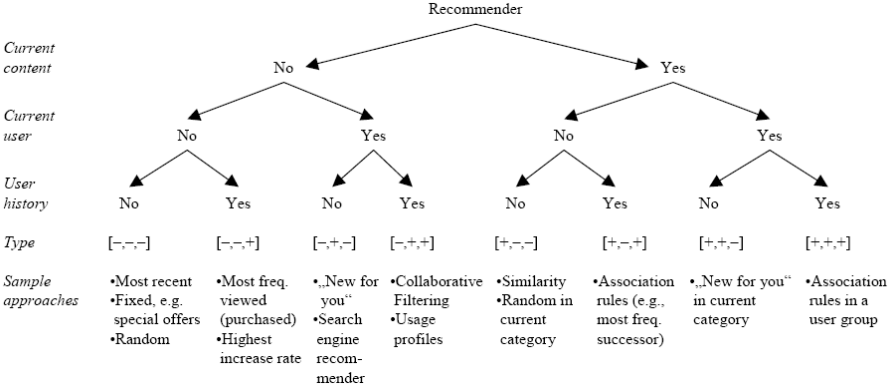
Recommenders

Recommender classification

- A general top level classification of website recommenders, focusing on context information is proposed
- The classification helps to compare different recommenders and guides in the evaluation of different approaches
- Three binary criteria are used to classify recommenders:
 - ▶ Current content
 - ▶ Current user
 - ▶ User history
- This leads to a distinction of 8 types of recommenders
- Each recommender type is represented by a three-character-code

Recommenders

Recommender classification



Recommenders

Example

- Type $[+,+,-]$ holds for recommenders that use information on the current content and current user, but not user history
- Type $[+,-,-]$ is to recommend content that is most similar to the current content (e.g. text-based similarity metrics)
- Type $[-,-,-]$ holds for content-insensitive recommenders for most recent content
- Type $[-,+,-]$ for users coming from a search engine like Google (SER - Search Engine Recommender)
- Type $[-,-,+]$ based on previous navigation patterns of website users, recommends the most frequently purchased/viewed content or with the highest increase of interest
- Type $[+,-,+]$ Association rule based recommenders ("Users who bought this item also bought. . .")

Recommender evaluation

Intro

- Metrics for measuring recommendation quality
- Evaluation for a sample non-commercial website

Recommender evaluation

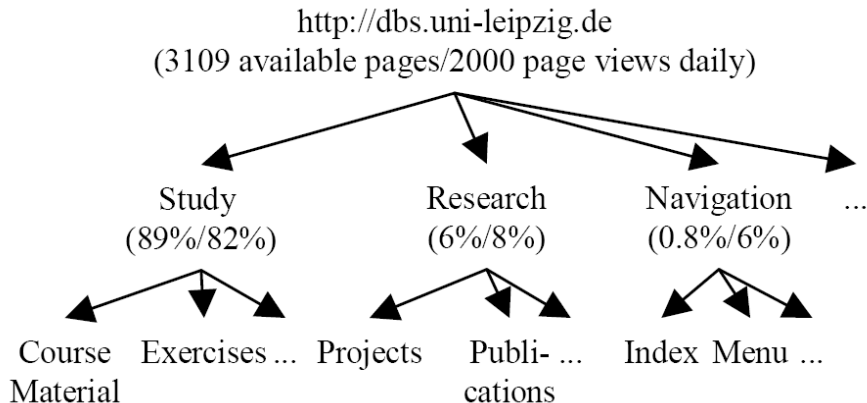
Evaluation metrics

- Accepted, Viewed and Purchased measures are used to evaluate the quality of presented recommendations
- Metrics are differentiated at two levels of granularity:
 - ▶ page views level
 - ▶ user sessions level
- The adopted metrics are:
 - ▶ $AcceptanceRate = |P_A|/|P|$
 - ▶ $SessionAcceptanceRate = |S_A|/|S|$
 - ▶ $ViewRate = |P_V|/|P|$, where $P_A \subseteq P_V \subseteq P$
 - ▶ $SessionViewRate = |S_V|/|S|$, where $S_A \subseteq S_V \subseteq S$
 - ▶ $RecommendedPurchaseRate = |S_{AP}|/|S|$, where $S_{AP} \subseteq S_A \subseteq S$

Recommender evaluation

Sample evaluation

- Database Group website (<http://dbs.uni-leipzig.de>)
- 4000 recommendations are presented every day
- 2 months evaluation



Recommender evaluation

Sample evaluation

- Only the results of six representative recommenders are presented

Recommender	
Type	Name
$[-,-,-]$	Most recent
$[-,-,+]$	Most frequent
$[-,+,-]$	SER
$[-,+,+]$	Personal Interests
$[+,-,-]$	Similarity
$[+,-,+]$	Association Rules

Recommender evaluation

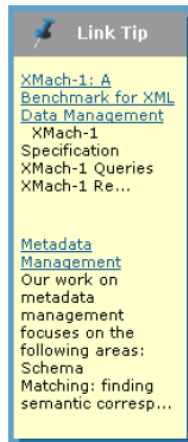
Sample evaluation

- Primarily, the AcceptanceRate metric has been used for the evaluation
- Average acceptance rate for all considered recommenders: 1.34%
- Average view rate: 14.54%
- Average session acceptance rate: 8.16%
- Session view rate: 25.24%

Recommender evaluation

Recommendations

- Low acceptance rates are influenced by the following facts:
 - ▶ every single web page contains a full navigation menu with 78 links
 - ▶ the recommendations are not highlighted using big fonts



Recommender evaluation

Sample website

Welcome to the database group at the University of Leipzig | Database Group Leipzig - Windows Internet Explorer

http://dbz.uni-leipzig.de/en/frontpage

Database Group Leipzig
within the department of computer science

UNIVERSITÄT LEIPZIG

search

Staff Research Study Service Shortcuts

Hot Links

- Arbeitskreis "Web und Datenbanken"
- Datenbanksysteme 2 (E. Rahm)
- DBSI Wh-Klausur Dez06
- Implementierung von Datenbanksystemen I (E. Rahm)
- Datenschutz und Datensicherheit (D. Sosna)
- Klausuren

Date tracker

- 14.12: Datenbanksysteme 1 für Wirtschaftsinformatiker (D. Sosna)
- 15.12: Datenschutz und Datensicherheit (D. Sosna)
- 15.12: Datenbankpraktikum WS 2006/07
- 15.12: News

Home

Welcome to the database group at the University of Leipzig

Prof. Dr. Erhard Rahm

News

Krankheitsbedingt müssen die Lehrveranstaltungen von Herrn Dr. Sosna in der Woche vom 11.-15.12.2006 leider ausfallen.

Research

- Publications (2006, 2005, 2004, 2003, ...)
- P2P data integration (iFuice, MOMA)
- Schema Management (COMA++)
- Online bibliography on Schema Evolution

Study

E-Learning
Lernmaterialien
Module der Abteilung WS 2006/07
Vergangene Semester
Klausuren
Abschlussarbeiten
LOTS-Nutzung
Erasmus

Abgeschlossene Arbeiten
Oberseminare
Themen
Anfrage SHK-Fähigkeit
Anfrage Abschlussarbeit

COMA++

Internet 100%

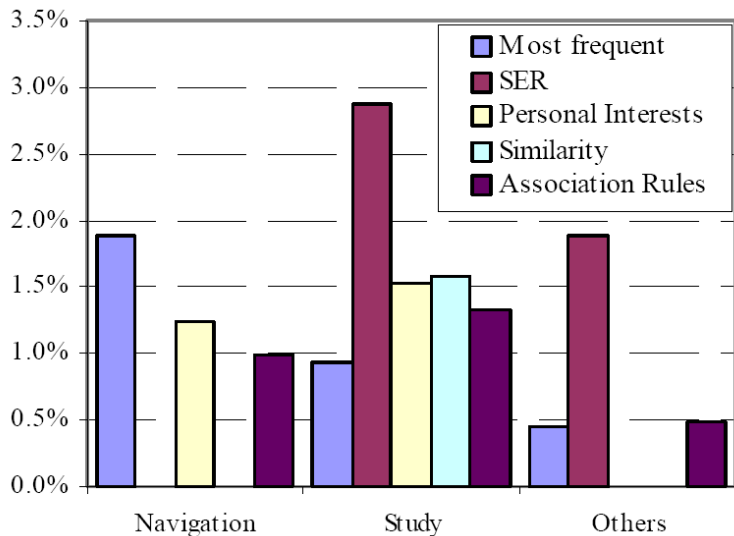
Recommender evaluation

Acceptance rates

Recommender		User type		
Type	Name	New users	Returning users	Σ
$[-,-,-]$	Most recent	(0.42%)	(0.00%)	(0.38%)
$[-,-,+]$	Most frequent	1.00%	0.62%	0.92%
$[-,+,-]$	SER	2.84%	1.95%	2.79%
$[-,+,+]$	Personal Interests	–	1.54%	1.54%
$[+,-,-]$	Similarity	1.65%	0.82%	1.56%
$[+,-,+]$	Association Rules	1.16%	0.68%	1.08%
	Σ	1.82%	1.09%	1.69%

Recommender evaluation

Recommenders comparison



Rule-based Recommender Selection

AWESOME supports

- Dynamic selection of recommender for every web site access
- Selection rules can be manually specified or generated automatically

Selection Rules

Structure of the selection rule

ContextPattern \Rightarrow recommender[weight]

Example of Rules

Context pattern		Recommender	Weight
Usertype="new user" AND ContentCategory1="Navigation"	\Rightarrow	"Most frequent"	[0.6]
Referrer="search engine"	\Rightarrow	"SER"	[0.8]
Clienttype="university" AND Usertype="returning user"	\Rightarrow	"Personal interest"	[0.4]

- Context pattern is a sequence of values from different context attributes.
- Recommender is the specific implementation of recommendation algorithm
- Weight $\in [0, 1]$ measures how good each recommender fits to each web access

Selection Rules (Continued)

Benefits of such system design

- Can be stored and retrieved from DBMS, that allows to store and query a lot of rules. **Not a common practice.**
- Allow a straight-forward and efficient implementation of recommender selection
- Rules can be generated off-line
- Manual insertion of rule is possible
- Selection criteria is flexible.
 - ▶ In paper use only MAX selection
 - ▶ Can use AVG, since many rules can lead to one recommender

Generating Selection Rules

Approaches

- Query-based top recommender
- Machine-learning approach

Query-based Top Recommender

- Takes advantage of warehouse query functionality.

Algorithm

- 1 Find all relevant context patterns in the recommendation fact table, i.e. context patterns exceeding a minimal support
- 2 For every such context pattern P do
 - ▶ Find recommender R with highest acceptance rate A
 - ▶ Add selection rule $P \Rightarrow R [A]$
- 3 Delete inapplicable rules

Query-based Top Recommender (Example)

- Find all relevant context patterns in the recommendation fact table, i.e. context patterns exceeding a minimal support
- For every such context pattern P do
 - ▶ Find recommender R with highest acceptance rate A
 - ▶ Add selection rule $P \Rightarrow R [A]$
- Delete inapplicable rules

Table example (simplified schema)

Usertype	ContentCategory	...	Referrer	Clienttype
"new user"	"Navigation"	...	"internal site"	"university"
...				

Query example to generate rules

```
SELECT Usertype, ..., Referrer, Clienttype, COUNT(*)  
FROM recommendations  
GROUP BY CUBE (Usertype, ..., Referrer, Clienttype)  
ORDER BY ...
```

Query-based Top Recommender (Example)

- Find all relevant context patterns in the recommendation fact table, i.e. context patterns exceeding a minimal support
- For every such context pattern P do
 - ▶ Find recommender R with highest acceptance rate A
 - ▶ Add selection rule $P \Rightarrow R [A]$
- Delete inapplicable rules

Returned context patterns example

Usertype	ContentCategory	...	Referrer	Clienttype	COUNT(*)
"new user"	"Navigation"	...	"internal site"	"university"	2
"returning user"	"Navigation"	...	"internal site"	"university"	3
NULL	"Navigation"	...	"internal site"	"university"	5
...					
NULL	NULL	...	"internal site"	"university"	40
...					
NULL	NULL	...	NULL	NULL	11432

Leave only significant rules (e.g. $HAVING COUNT(*) > 100$)

Query-based Top Recommender (Example)

- Find all relevant context patterns in the recommendation fact table, i.e. context patterns exceeding a minimal support
- For every such context pattern P do
 - ▶ Find recommender R with highest acceptance rate A
 - ▶ Add selection rule $P \Rightarrow R [A]$
- Delete inapplicable rules

Example

Usertype="returning user" AND ContentCategory1="Navigation"

"Most frequent"	[0.06]
"Personal interest"	[0.02]
"SER"	[0.01]

Will generate rule :

Usertype="returning user" AND ContentCategory1="Navigation" \Rightarrow "Most frequent" [0.06]

Query-based Top Recommender (Example)

- Find all relevant context patterns in the recommendation fact table, i.e. context patterns exceeding a minimal support
- For every such context pattern P do
 - ▶ Find recommender R with highest acceptance rate A
 - ▶ Add selection rule $P \Rightarrow R [A]$
- Delete inapplicable rules

Example

Usertype="returning user" AND ContentCategory1="Navigation"	\Rightarrow	"SER"	[0.01]
Usertype="returning user"	\Rightarrow	"Most frequent"	[0.06]

Rule Usertype="returning user" AND ContentCategory1="Navigation" \Rightarrow "SER" [0.01] **will be deleted.**

Machine-learning Approach

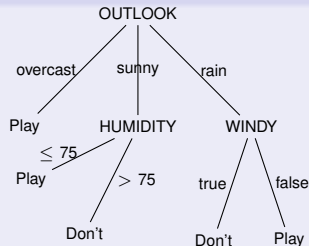
- Use Weka J48 algorithm to induce the rules from the data
- J48 is C4.5 Java implementation from Weka project
- C4.5 itself is ID3 (studied in DWDM course) modification
 - ▶ deals with unknown attribute values
 - ▶ deals with attributes with continuous ranges
 - ▶ use pruning of decision trees

Short Reminder of ID3 and C4.5

When to play golf? Classical example.

Example of data and corresponding classification tree

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play



Data Preparation for Machine-learning Approach

Data example (Only few attributes)

attribute 1	attribute 2	attribute 3	Recommender	Accepted recommendation
"returning user"	"search engine"	"Navigation"	"Most frequent"	true
"returning user"	"search engine"	"Navigation"	"Most frequent"	false
"returning user"	"search engine"	"Navigation"	"SER"	false
"new user"	"search engine"	"Navigation"	"Most frequent"	true
...				

- Take context patterns with **fully specified** attributes which exceeds threshold.
 - ▶ 3 page views with content pattern {"returning user", "search engine", "Navigation"}
 - ▶ for recommender "Most frequent"
 - ▶ 1 accepted recommendation (out of 3) $1/3 = 0.333$
 - ▶ $k=10$
 - ▶ Add 3 ($10 \cdot 0.3$) occurrences to training data with such content pattern and recommender "Most frequent" as a class
- Bigger acceptance rates more training instances

From Data to Recommendation rules

- Run Weka J48 on prepared data with **every recommender as a class**
- Extract rules from generated decision tree
 - ▶ Every path from the root to a leaf defines a context pattern
 - ▶ Unspecified context attributes are set to NULL (accept any)
 - ▶ Weight of the rule is set to the relative fraction of correctly classified instances
 - ★ Classify all data with generated decision tree to get the fractions

Evaluation of Selection Rules (1/3)

Tested selection strategies

Name	Description
Top-Rec	Automatic strategy of section 5.2.1 (query-based)
Decision Tree	Automatic strategy of section 5.2.2 (machine learning)
Manual 1	The most frequently viewed pages per content category are recommended. For returning users, this category is derived from previous sessions of the current user. For new users, the category of the current page is used.
Manual 2	For search engine users, the search engine recommender is applied. Otherwise the content similarity recommender (for course material pages) or association rule recommender (for other pages) is selected.
Random	Random selection of a recommender

- 2 automatically generated selection rules, 2 manually inserted, and 1 random
 - ▶ **Manual 1:** most frequent per category is recommended
 - ▶ **Manual 2:** utilizes background knowledge about the web site structure and typical user groups as well as evaluation results obtained after an extensive manual OLAP analysis
- For every user session one strategy was randomly selected
- Two months of data

Evaluation of Selection Rules (2/3)

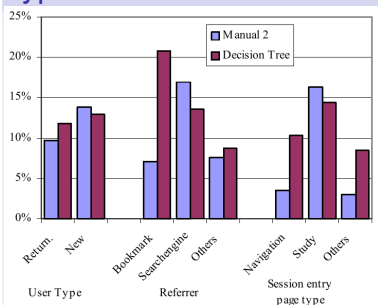
Comparison of selection strategies

Strategy	Nr. of rules	Selection time	Acceptance rate	Session acceptance rate
Top-Rec	~ 2000	~ 14 ms	1.25 %	9.58 %
Decision Tree	~ 250	~ 12 ms	1.64 %	12.54 %
Manual 1	24	~ 13 ms	0.96 %	7.11 %
Manual 2	5	~ 13 ms	1.84 %	12.47 %
Random	137	~ 19 ms	0.89 %	6.51 %

- Moderate number of rules
- Fast recommender selection
- Decision Tree as good as manual rules, but completely automatic
- Decision Tree is better than query based top recommender approach
 - ▶ Most significant attributes has big influence on selection process

Evaluation of Selection Rules (3/3)

Acceptance rate wrt user type, referrer and session entry page type



- Measure session acceptance rate
- Manual rules are better for search engine referred users
- Overall decision tree generated rules are as good as manually generated
- Adding several manual rules to automatically generate, would improve overall performance

Conclusions

- New data warehouse based web site evaluation and recommendation system
 - ▶ Allows coordinated use of large number of recommenders
 - ▶ Automatically selects best promising recommender (as good as manual selection)
- What could be improved
 - ▶ Fine-tuning of the recommendation algorithm itself (similarity metrics, k-neighborhood search)
 - ▶ More sophisticated evaluation (not only acceptance rates) similar to [Agichtein et al., 2006]