

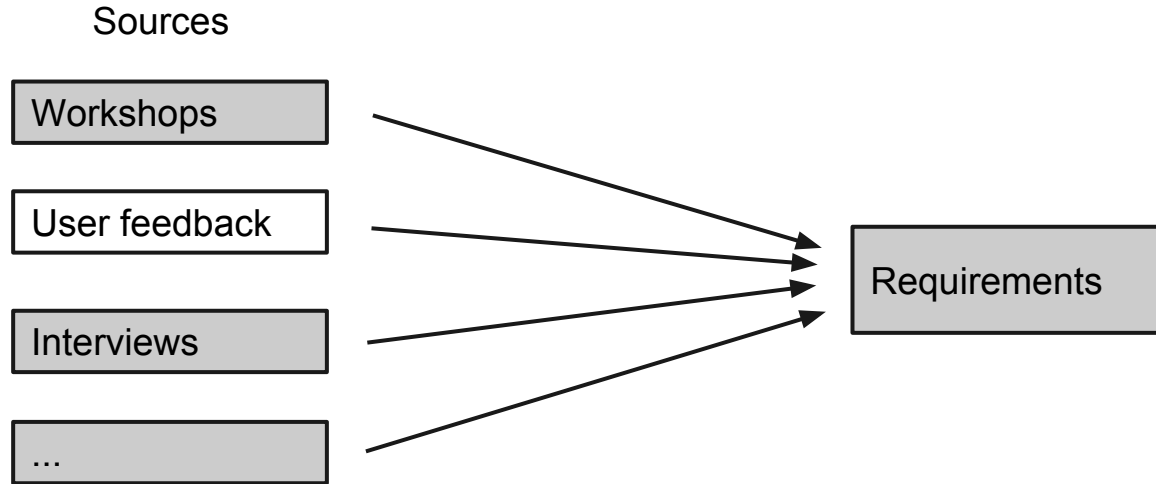
User Feedback for Continuous Software Evolution

Group 3 - RE II - HS14

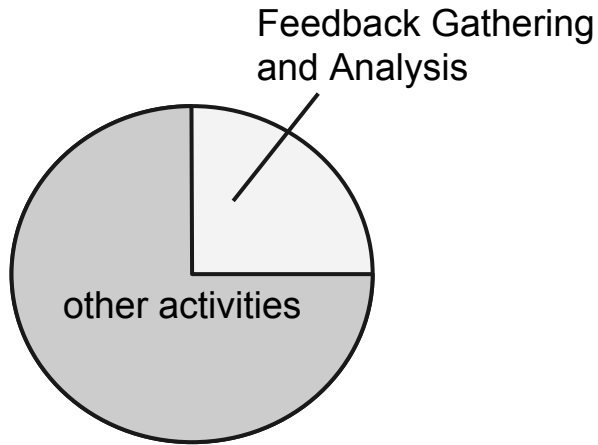
Agenda

- 1) How our topics fit in the context of mandatory readings
- 2) User Feedback in the AppStore
- 3) Analysis of User Comments: An approach for Software Requirements Evolution
- 4) Discussion

How our topic fits



User Feedback in the AppStore



stars

textual review

rating the helpfulness

Categorization

topic	description	freq.
praise	expresses appreciation	75.36%
helpfulness	scenario the app has proven helpful for	22.45%
feature information	concrete feature or user interface	14.45%
shortcoming	concrete aspect, user is not happy with	13.27%
bug report	bug report or crash report	10.00%
feature request	asks for missing feature	6.91%
other app	reference to other app, e.g. for comparison	3.91%
recommendation	suggests acquisition	3.82%
noise	meaningless information	3.27%
dissuasion	advises against purchase	3.27%
content request	asks for missing content	2.91%
promise	trades a better rating for a specific improvement	2.00%
question	asks how to use specific feature	1.27%
improvement request	requests improvement (e.g. app is slow)	1.18%
dispraise	opposite of praise	1.18%
other feedback	references or answers other feedback	1.09%
howto	explains other users how to use app	0.91%

Impact

- download numbers
- better ratings
- bug reports
- user experience
- feature requests

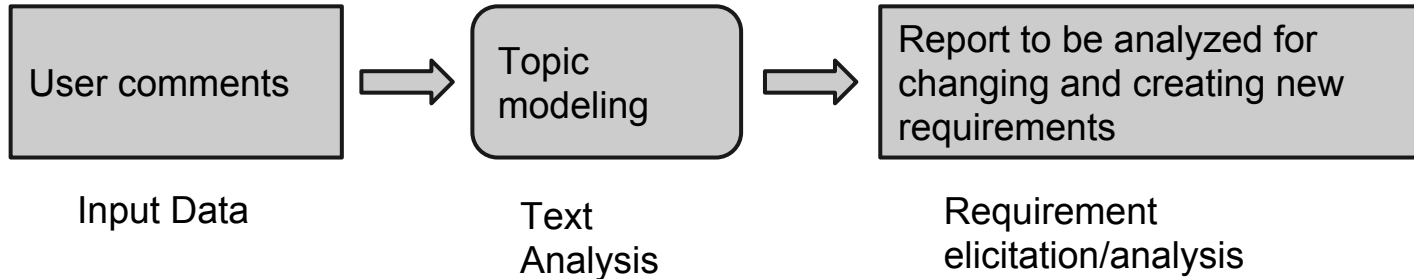
Difficulties

- No filter, aggregation, classification
- No usage or context data
- No bidirectional communication
- Quality varies widely
- Large number of reviews

Analysis of User comments

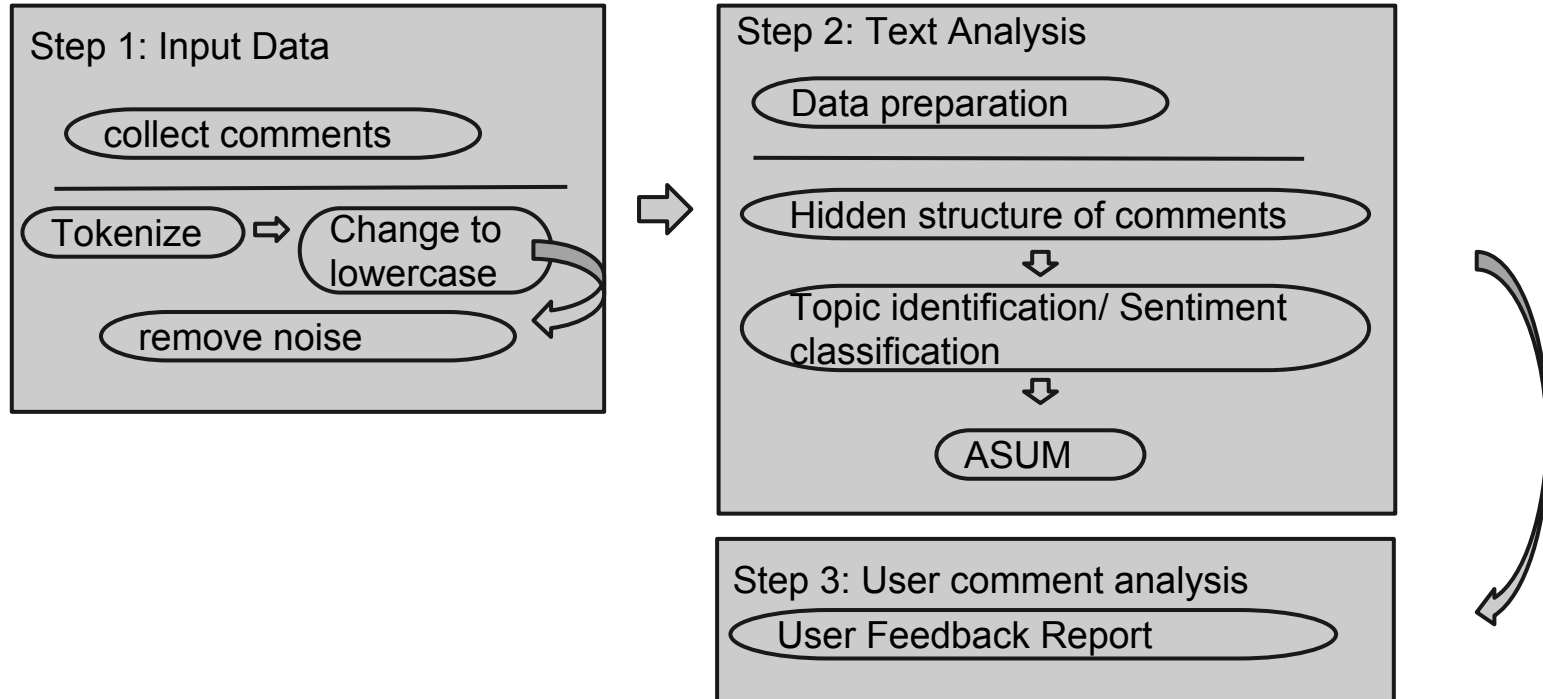
- User feedback = main driver for software evolution
- Requirement changes due to extracted information
 - ➡ by focus groups, surveys, error reports, comments on sites
- Examples: Android Marketplace, App Store
- Problem: 1) volume of data 2) time commitment involved in extracting requirements
- Automated analysis technique of user feedback

High-level overview



- Automatically extracting topics for changing and creating new requirements
- Extracting topics prevalent in user comments
- Evaluation:
 - Compare manual/automated effort/accuracy of extraction
 - reasoning about possible requirement changes

3 steps solution



Evaluation of Approach



Mint.com
Personal
Finance



Calorie
Tracker



Facebook for
Android

Results



Calorie
Tracker

Results achieved
through user
comments analysis:

- more efficient
- easily find nonsense sentences
- small amount of time for extracting informations (compared to manual process)

- Barcode scanner
- Calorie counter
- Database
- Not enough features
- Creating an account
- Sign in/ log in
- Bad interface
- Wrong visualization of food
- Difficult to use
- Break down of nutrition

Difficulties

However the approach shows some weaknesses...

- large number of topics to avoid mis-classifications → what if there are not a lot of comments for an application?
- Scope of topics still high? e.g. Facebook 1941 comments → 100-200 topics
- Cluster Analysis: Consideration of center sentence of each cluster → mis-classifications

Our Experience - Social Computing for RE

- Market-oriented mobile app
- No explicit use
- Release information published on Twitter and Facebook
- User comments

Discussion

- What are alternative approaches to gather feedback for the evolution of mobile apps?
- Do you know other business fields where the approach of user comments analysis could be applied?