Quality Assurance for Human Computation Based Recommendation

Master Defense Presentation

Michael Schwarz

June 13, 2017

Graz University of Technology

Introduction

Motivation

- Recommender systems are ubiquitous

Motivation

- Recommender systems are ubiquitous
- These systems are usually based on knowledge

Motivation

- Recommender systems are ubiquitous
- These systems are usually based on knowledge
- Reliable but expensive if entered by small number of experts

Motivation

- Recommender systems are ubiquitous
- These systems are usually based on knowledge
- Reliable but expensive if entered by small number of experts
- Unreliable but cheap if entered by regular users

Motivation

- Recommender systems are ubiquitous
- These systems are usually based on knowledge
- Reliable but expensive if entered by small number of experts
- Unreliable but cheap if entered by regular users
- Combine approaches to reliably and cheaply collect knowledge

Overview

- Design and implement a web-based generic recommender platform

Overview

- Design and implement a web-based generic recommender platform
- Add mechanisms to collect data from regular users

Overview

- Design and implement a web-based generic recommender platform
- Add mechanisms to collect data from regular users
- Develope techniques to ensure the quality of the collected data

Overview

- Design and implement a web-based generic recommender platform
- Add mechanisms to collect data from regular users
- Develope techniques to ensure the quality of the collected data
- Efficiently distribute tasks to users to improve the knowledge base

Recommender Systems

Recommender Systems

- There are different types of recommender systems

Recommender Systems

- There are different types of recommender systems
- They all recommend products/items...

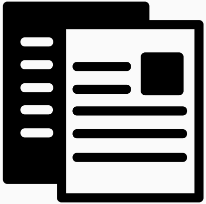
Recommender Systems

- There are different types of recommender systems
- They all recommend products/items...
- ...but use different techniques to find the best item(s)

Recommender Systems

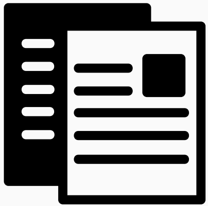
- There are different types of recommender systems
- They all recommend products/items...
- ...but use different techniques to find the best item(s)
- Three types of systems are commonly used

Content-based Systems



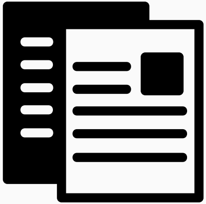
- Collect information about the items (e.g., keywords)

Content-based Systems



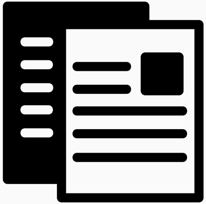
- Collect information about the items (e.g., keywords)
- Find items similar to ones the user liked in the past

Content-based Systems



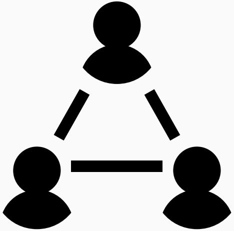
- Collect information about the items (e.g., keywords)
- Find items similar to ones the user liked in the past
- Idea: user preferences do not change

Content-based Systems



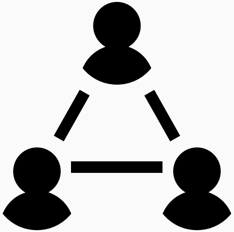
- Collect information about the items (e.g., keywords)
- Find items similar to ones the user liked in the past
- Idea: user preferences do not change
- Advantage: independent of other users

Collaborative Filtering Systems



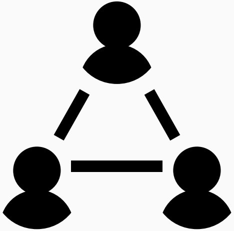
- Collect information about the user

Collaborative Filtering Systems



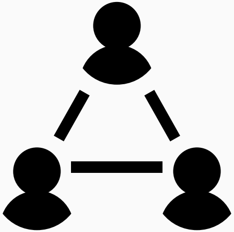
- Collect information about the user
- Find similar users

Collaborative Filtering Systems



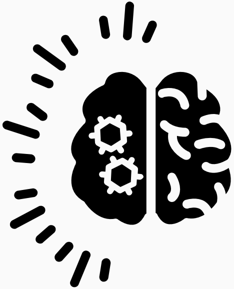
- Collect information about the user
- Find similar users
- Idea: people who liked the same things will like the same in the future

Collaborative Filtering Systems



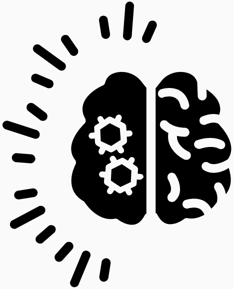
- Collect information about the user
- Find similar users
- Idea: people who liked the same things will like the same in the future
- Advantage: no understanding of the items necessary

Knowledge-based Systems



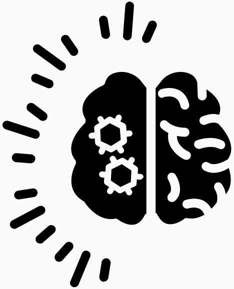
- Explicit information about the items and user

Knowledge-based Systems



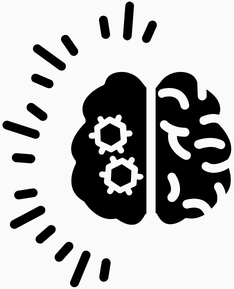
- Explicit information about the items and user
- Find items that fulfill the user-given constraints

Knowledge-based Systems



- Explicit information about the items and user
- Find items that fulfill the user-given constraints
- Idea: recommendation boils down to a constraint satisfaction problem

Knowledge-based Systems



- Explicit information about the items and user
- Find items that fulfill the user-given constraints
- Idea: recommendation boils down to a constraint satisfaction problem
- Advantage: no history of the user is necessary

A Generic Framework

Web-based Client-Server Model

- Subdivided into frontend and backend

Web-based Client-Server Model

- Subdivided into frontend and backend
- Backend is based on the Spring Framework (Java)

Web-based Client-Server Model

- Subdivided into frontend and backend
- Backend is based on the Spring Framework (Java)
- Frontend is mobile-friendly HTML5

Web-based Client-Server Model

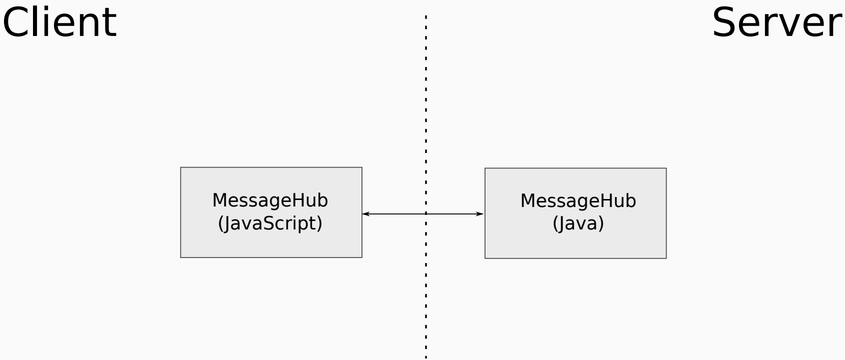
- Subdivided into frontend and backend
- Backend is based on the Spring Framework (Java)
- Frontend is mobile-friendly HTML5
- Parts are loosely coupled

Message Passing

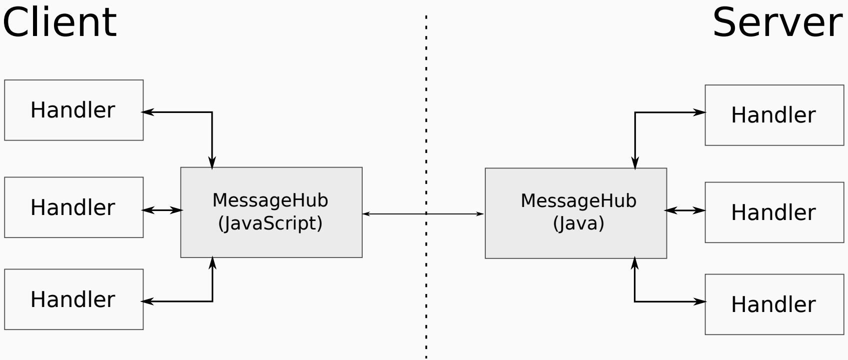
Client

Server

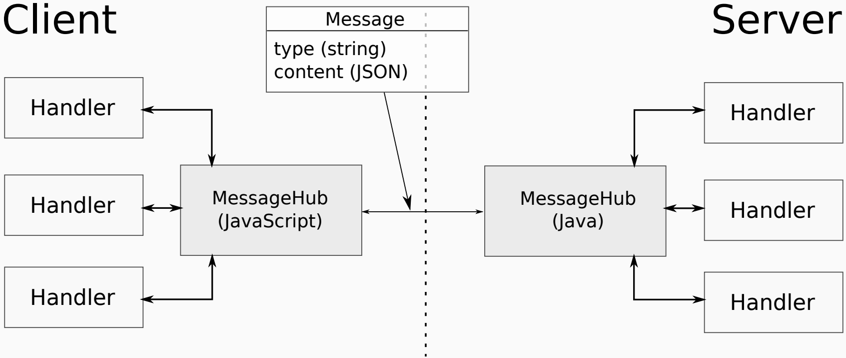
Message Passing



Message Passing



Message Passing



Message Format

Message to register a new user

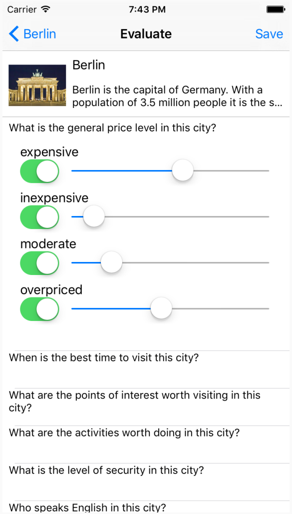
```
{
  type: "register",
  content : {
    username: "michael",
    password: "12345678",
    email: "michael.schwarz@noreply.com"
  }
}
```

Multiple Frontends

- Loose coupling and easy API allows easy implementation of new frontends

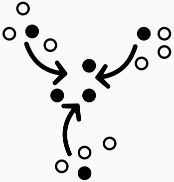
Multiple Frontends

- Loose coupling and easy API allows easy implementation of new frontends
- Bachelor Thesis: Implementation of a native iOS client



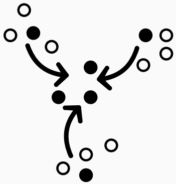
Knowledge Acquisition

Acquire Knowledge



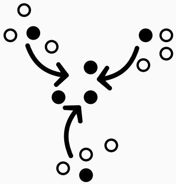
- Users do not like lengthy tasks

Acquire Knowledge



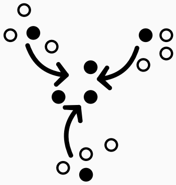
- Users do not like lengthy tasks
- Acquire knowledge from the user using small tasks (microtasks)

Acquire Knowledge



- Users do not like lengthy tasks
- Acquire knowledge from the user using small tasks (microtasks)
- Microtask has only one question

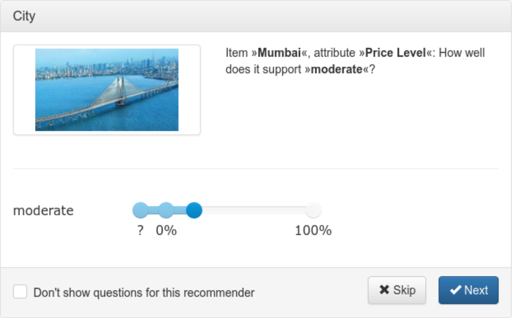
Acquire Knowledge



- Users do not like lengthy tasks
- Acquire knowledge from the user using small tasks (microtasks)
- Microtask has only one question
- 6 different types of microtasks

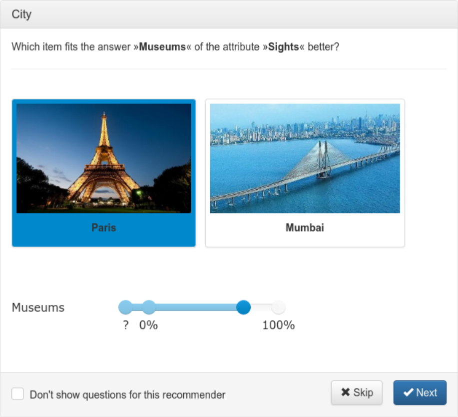
Microtask #1

Item’s support regarding one specific attribute



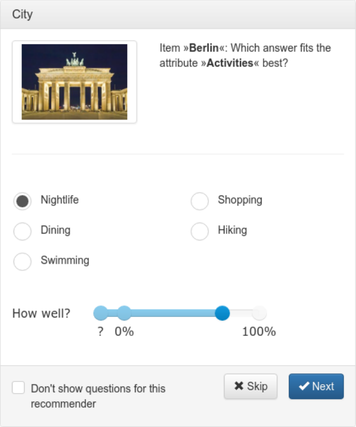
Microtask #2

Best matching item regarding one specific attribute



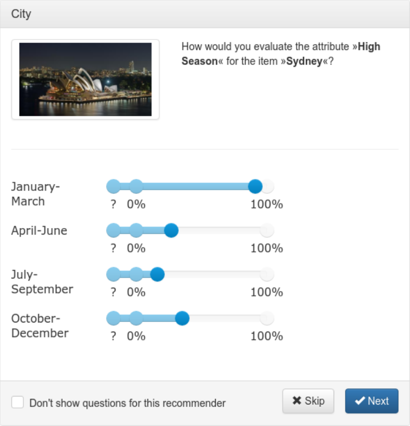
Microtask #3

Best matching answer regarding one specific attribute



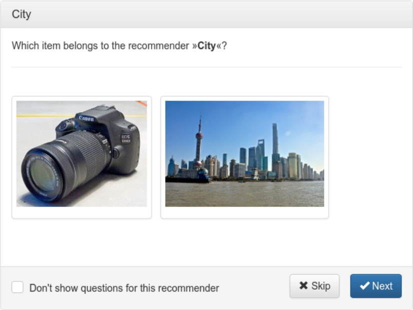
Microtask #4

Weighted answers regarding one specific attribute



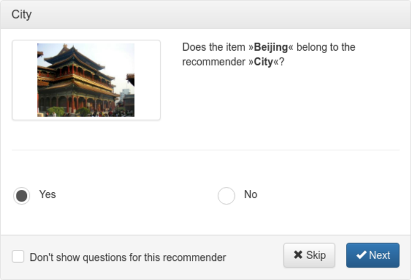
Microtask #5

Implicit CAPTCHA



Microtask #6

Binary decision



Quality Assurance

Human Score

- Users have to “earn” trust

Human Score

- Users have to “earn” trust
- Score is influenced by CAPTCHAs, user behavior, etc.

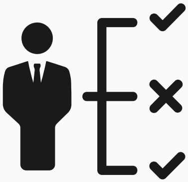
Human Score

- Users have to “earn” trust
- Score is influenced by CAPTCHAs, user behavior, etc.
- All contributions of the user are weighted with this score (0 % - 100 %)

Human Score

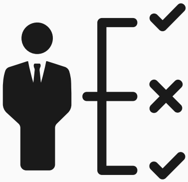
- Users have to “earn” trust
- Score is influenced by CAPTCHAs, user behavior, etc.
- All contributions of the user are weighted with this score (0 % - 100 %)
- New or malicious users have minor to no influence on the knowledge base

Ground Truth



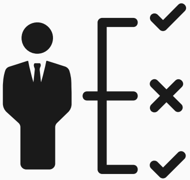
- Depending on the human score, users get microtask with known answers (ground truth)

Ground Truth



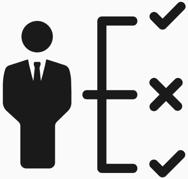
- Depending on the human score, users get microtask with known answers (ground truth)
- Similar to CAPTCHAs, but not seen as such by the user

Ground Truth



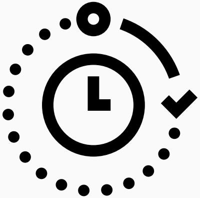
- Depending on the human score, users get microtask with known answers (ground truth)
- Similar to CAPTCHAs, but not seen as such by the user
- Influence the human score (positively and negatively)

Ground Truth



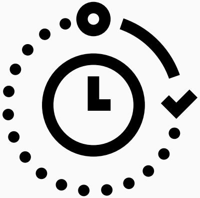
- Depending on the human score, users get microtask with known answers (ground truth)
- Similar to CAPTCHAs, but not seen as such by the user
- Influence the human score (positively and negatively)
- Classify an image, hard to do automatically

Timing Models



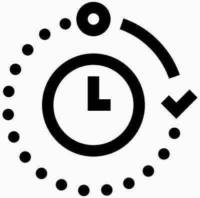
- We model the time it takes to answer a microtask

Timing Models



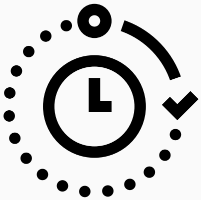
- We model the time it takes to answer a microtask
- Timings are matched using Kullback-Leibler distance

Timing Models



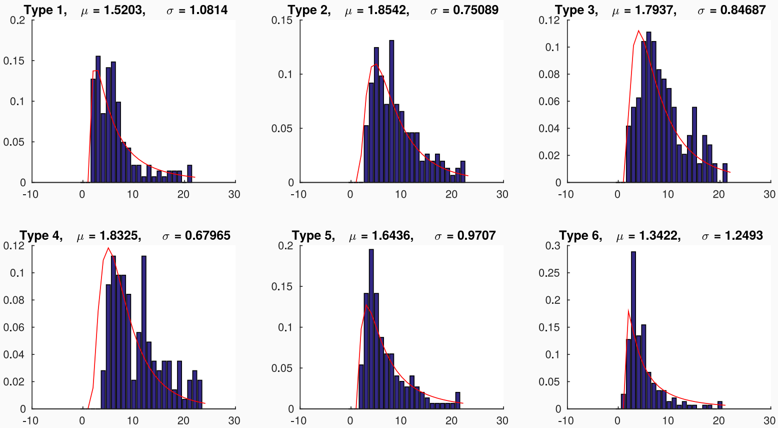
- We model the time it takes to answer a microtask
- Timings are matched using Kullback-Leibler distance
- Answers are weighted according to how well they fit

Timing Models



- We model the time it takes to answer a microtask
- Timings are matched using Kullback-Leibler distance
- Answers are weighted according to how well they fit
- Non-matching timings are discarded and decrease the human score

Microtask Timings



Spam

- Users can add new item, we have to cope with spam

Spam

- Users can add new item, we have to cope with spam
- CAPTCHAs only prevent automated spam

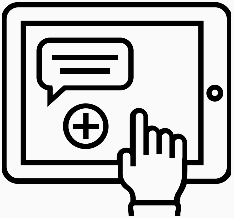
Spam

- Users can add new item, we have to cope with spam
- CAPTCHAs only prevent automated spam
- For a new item, we generate verification microtasks

Spam

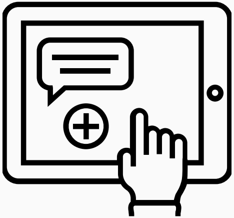
- Users can add new item, we have to cope with spam
- CAPTCHAs only prevent automated spam
- For a new item, we generate verification microtasks
- If the community decides that an item does not belong to the recommender, it is removed

Data Collection



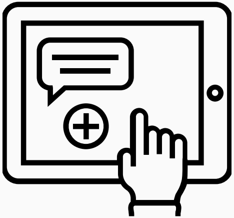
- We need knowledge for new items

Data Collection



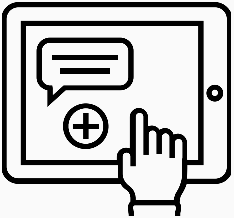
- We need knowledge for new items
- Dynamic approach to calculate number of distributed microtasks

Data Collection



- We need knowledge for new items
- Dynamic approach to calculate number of distributed microtasks
- Loosely based on local working set algorithm for task scheduling

Data Collection



- We need knowledge for new items
- Dynamic approach to calculate number of distributed microtasks
- Loosely based on local working set algorithm for task scheduling
- Settle on minimum number of microtasks based on quality of the results

Evaluation

Study

- We conducted a worldwide study

Study

- We conducted a worldwide study
- Users had to complete microtasks, evaluate items, and use the recommender

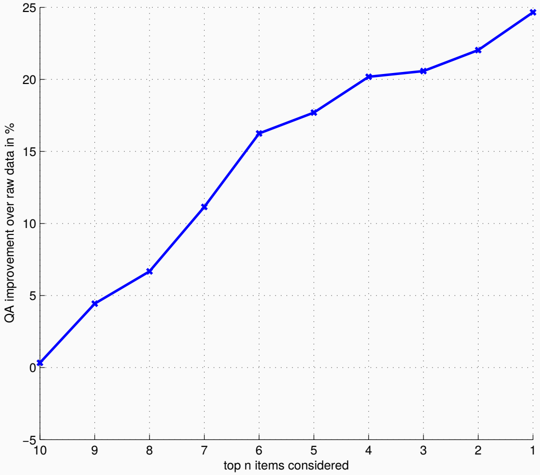
Study

- We conducted a worldwide study
- Users had to complete microtasks, evaluate items, and use the recommender
- 1307 users (90.9 %) completed all tasks

Study

- We conducted a worldwide study
- Users had to complete microtasks, evaluate items, and use the recommender
- 1307 users (90.9 %) completed all tasks
- Quality assurance led to recommendation improvement of >20 %

Recommendation Quality Improvement



Conclusion

Summary

- We developed a generic recommender framework for knowledge-based recommenders

Summary

- We developed a generic recommender framework for knowledge-based recommenders
- We showed that users are willing to contribute through small tasks

Summary

- We developed a generic recommender framework for knowledge-based recommenders
- We showed that users are willing to contribute through small tasks
- We presented automatic ways to ensure the quality of user content

Contribution #1

Human Computation Based Acquisition of Financial Service Advisory Practices

Alexander Felfernig, Michael Jeran, Martin Stettinger, Thomas Absenger, Thomas Gruber, Sarah Haas, Emanuel Kirchengast, Michael Schwarz, Lukas Skofitsch, Thomas Ulz

FINREC'15

Contribution #2

Peopleviews: Human computation for constraint-based recommendation

Alexander Felfernig, Thomas Ulz, Sarah Haas, Michael Schwarz, Stefan Reiterer, Martin Stettinger

ACM RecSys 2015 CrowdRec Workshop

Contribution #3

Human computation for constraint-based recommenders

Thomas Ulz, Michael Schwarz, Alexander Felfernig, Sarah Haas, Amal Shehadeh, Stefan Reiterer, Martin Stettinger

Journal of Intelligent Information Systems 2016

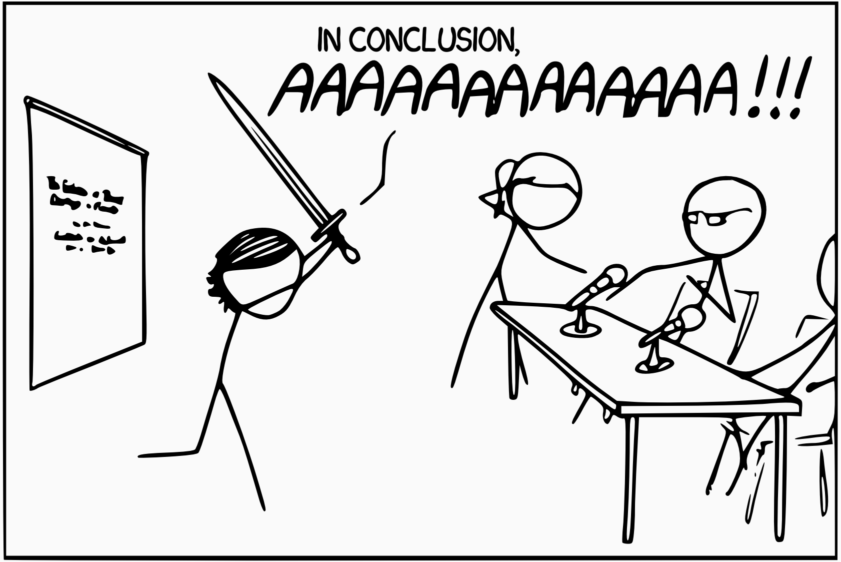
Contribution #4

A Short Overview of the PeopleViews Mobile User Interface

Angela Promitzer, Alexander Felfernig, Michael Schwarz, Thomas Ulz, Amal Shehadeh, Sarah Haas

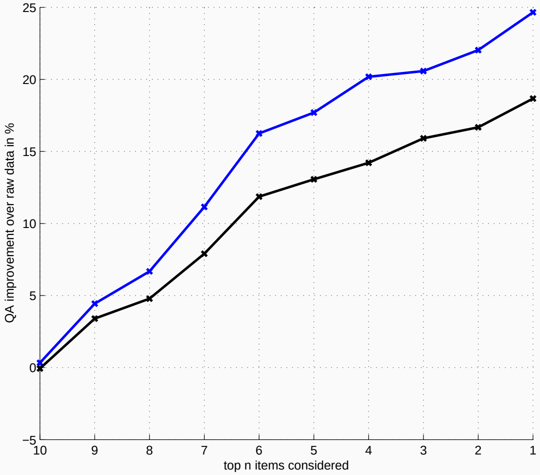
TU Graz Technical Report 2016

Thank you for your attention!



THE BEST THESIS DEFENSE IS A GOOD THESIS OFFENSE

Recommendation Quality Improvement without Ground Truth



Human Score Calculation Example

		Support			
User	Human Score	Answer 1	Answer 2	Answer 1 (weighted)	Answer 2 (weighted)
User 1	1	0.8	0.3	0.8	0.3
User 2	0.5	0.9	0.4	0.45	0.2
User 3	0.5	0.6	0.5	0.3	0.25
User 4	0	0.2	0.7	0	0
Sum	**2**	2.5	1.9	1.55	0.75
Average	-	$\frac{2.5}{4}$ = 0.625	$\frac{1.9}{4}$ = 0.475	$\frac{1.55}{\mathbf{2}}$ = 0.775	$\frac{0.75}{\mathbf{2}}$ = 0.375

Table 1: Four different users and their support values for Answer 1 and Answer 2.

Optimal Number of Microtasks Example

	# of microtasks	Answered	Data is good	New # of microtasks
Cycle 1	10	4	no	$10 \times 1.5 = 15$

Goal: 5 answers

Cycle 1 Start with 10 tasks $\rightarrow$ not enough, increase to 15

Optimal Number of Microtasks Example

	# of microtasks	Answered	Data is good	New # of microtasks
Cycle 1	10	4	no	$10 \times 1.5 = 15$
Cycle 2	15	11	yes	$15 \times 0.75 = 11$

Goal: 5 answers

Cycle 1 Start with 10 tasks $\rightarrow$ not enough, increase to 15

Cycle 2 15 was enough, decrase to 15 · 0.75 = 11 tasks

Optimal Number of Microtasks Example

	# of microtasks	Answered	Data is good	New # of microtasks
Cycle 1	10	4	no	$10 \times 1.5 = 15$
Cycle 2	15	11	yes	$15 \times 0.75 = 11$
Cycle 3	11	6	yes	$11 \times 0.75 = 8$

Goal: 5 answers

Cycle 1 Start with 10 tasks $\rightarrow$ not enough, increase to 15

Cycle 2 15 was enough, decrase to 15 $\cdot$ 0.75 = 11 tasks

Cycle 3 11 was enough, decrease to 11 $\cdot$ 0.75 = 6 tasks

Optimal Number of Microtasks Example

	# of microtasks	Answered	Data is good	New # of microtasks
Cycle 1	10	4	no	$10 \times 1.5 = 15$
Cycle 2	15	11	yes	$15 \times 0.75 = 11$
Cycle 3	11	6	yes	$11 \times 0.75 = 8$
Cycle 4	8	4	no	$8 \times 1.5 = 12$

Goal: 5 answers

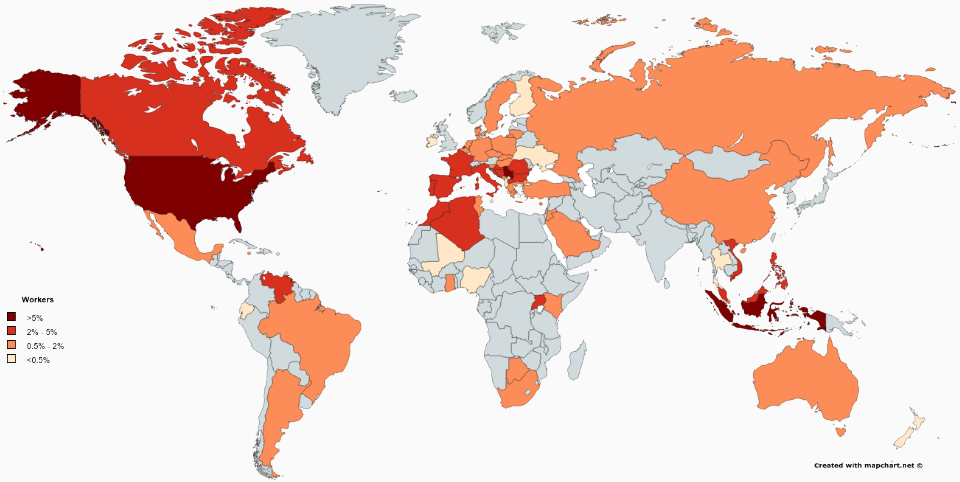
Cycle 1 Start with 10 tasks $\rightarrow$ not enough, increase to 15

Cycle 2 15 was enough, decrase to 15 · 0.75 = 11 tasks

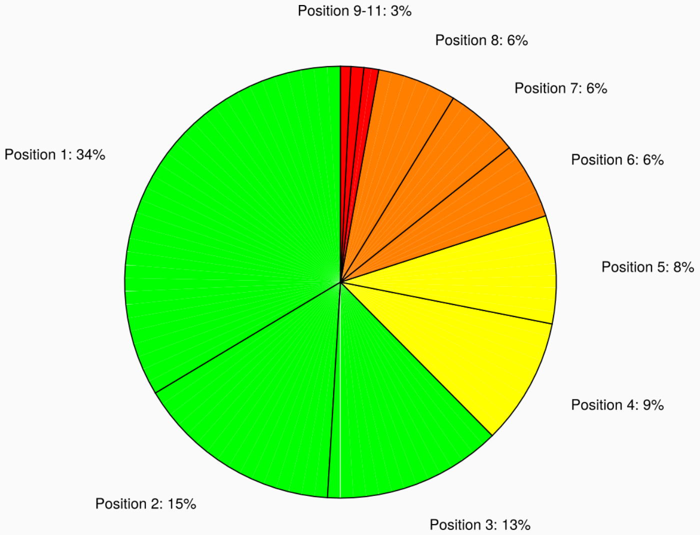
Cycle 3 11 was enough, decrease to 11 · 0.75 = 6 tasks

Cycle 4 8 was not enough, increase to 8 · 1.5 = 12 tasks

Worker Distribution



Position of Chosen Item



Recommendation Screen