STU FIIT

PeWe@FIIT
personalized web group

Personalized Text Summarization Based on Important Terms Identification

Róbert Móro, Mária Bieliková

Slovak University of Technology in Bratislava, Slovakia

robert.moro@stuba.sk

TIR ’12, 4.9.2012

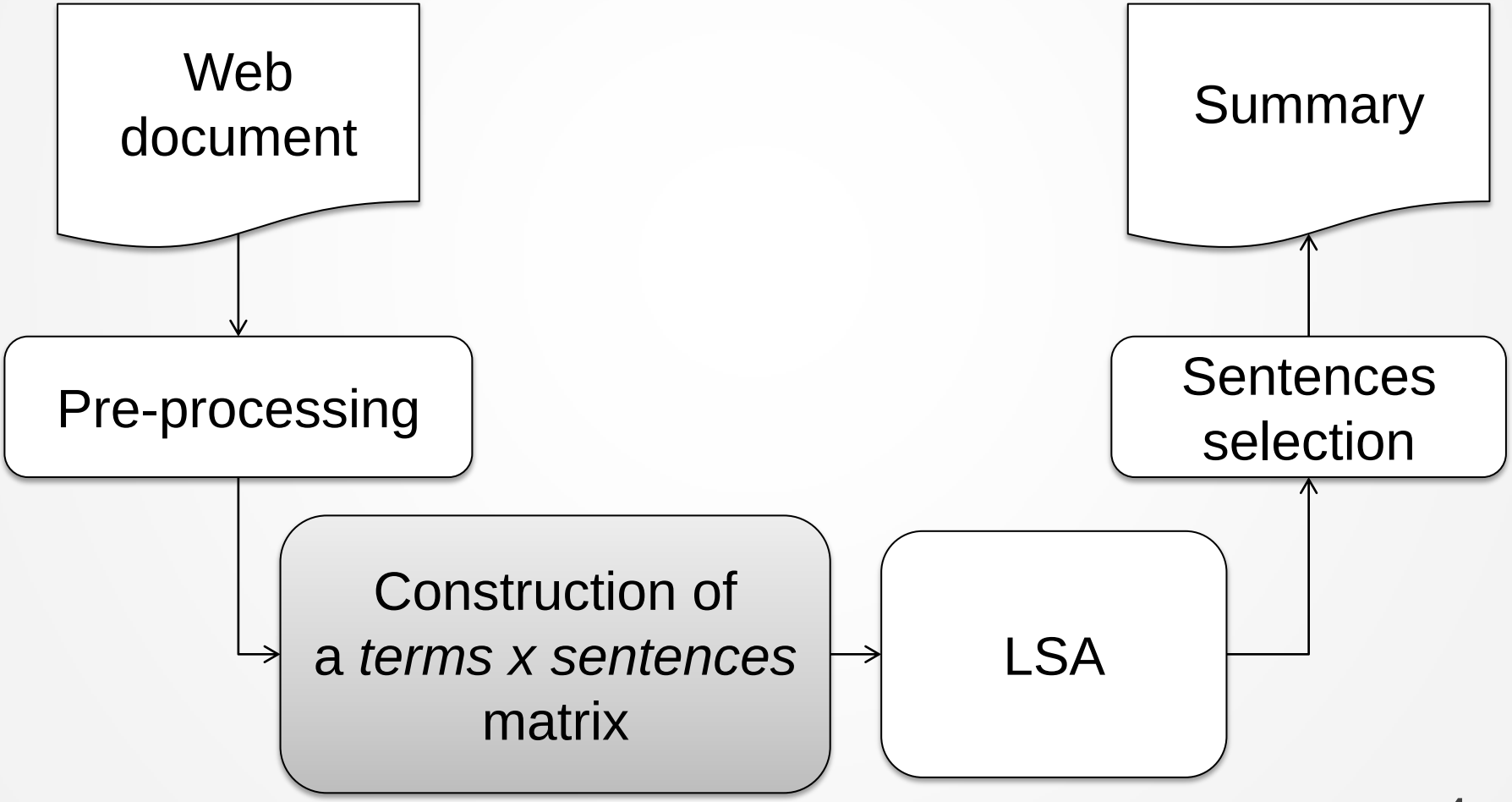
Motivation

- Information overload problem
  - Automatic text summarization
  - Personalization
- User characteristics
- Metadata
- Domain of learning
  - Summarization for revision

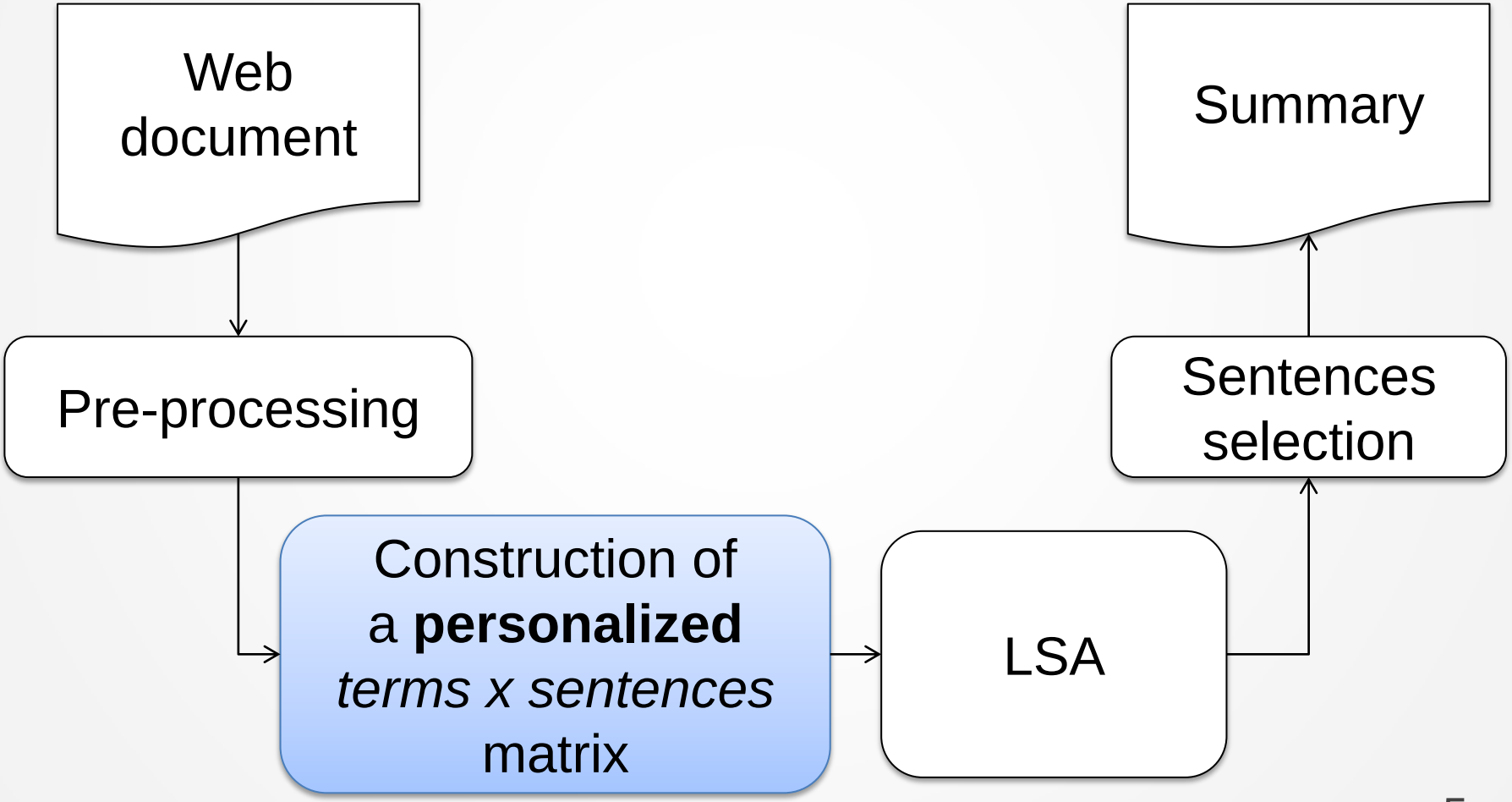
Method of Personalized Summarization

- Based on a method of latent semantic analysis
- Combination of different sources
  - Domain conceptualization
  - Knowledge of the users
  - Annotations (highlights) added by users
- Independent of the chosen domain

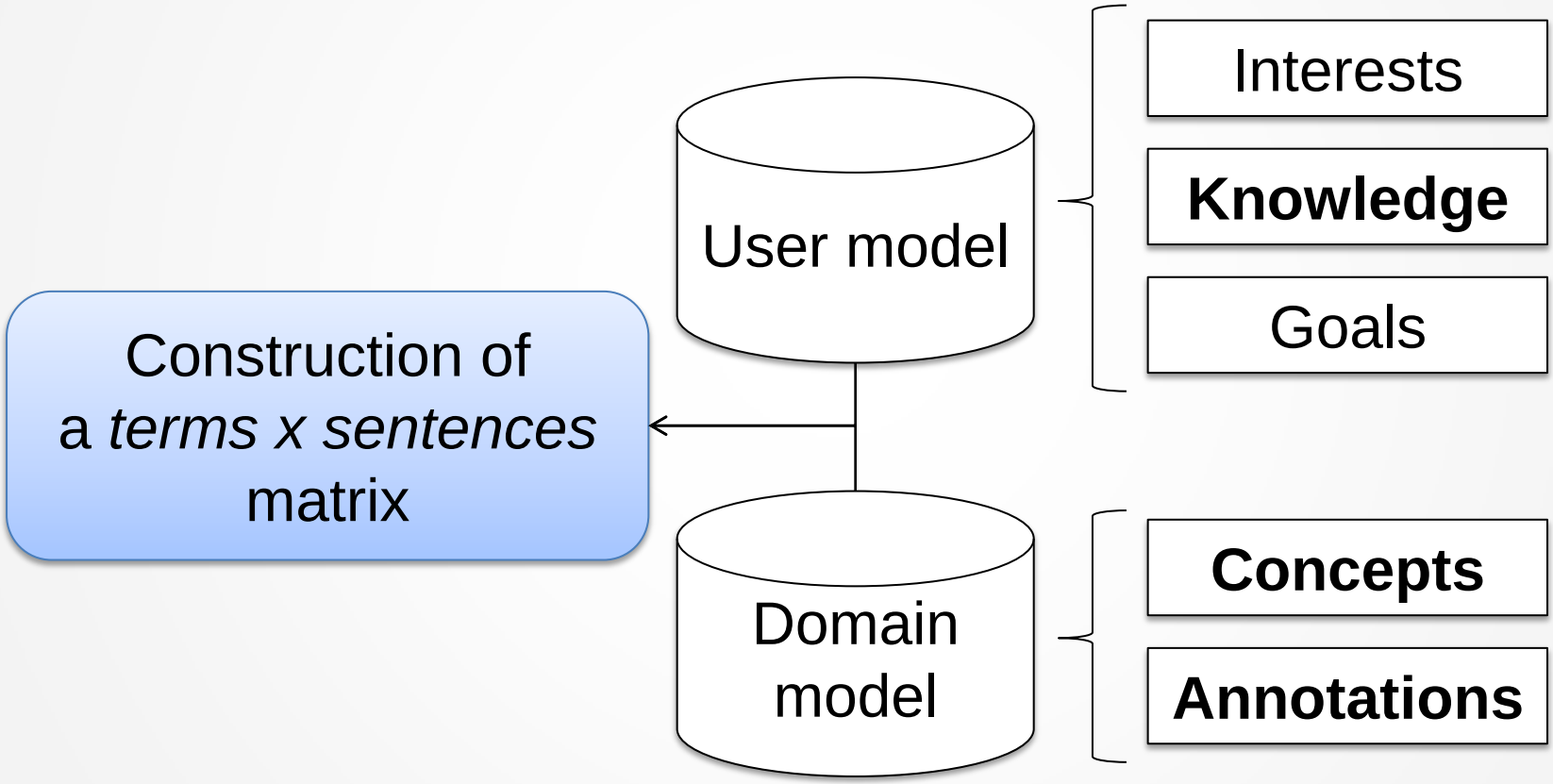
Method of Personalized Summarization



Method of Personalized Summarization



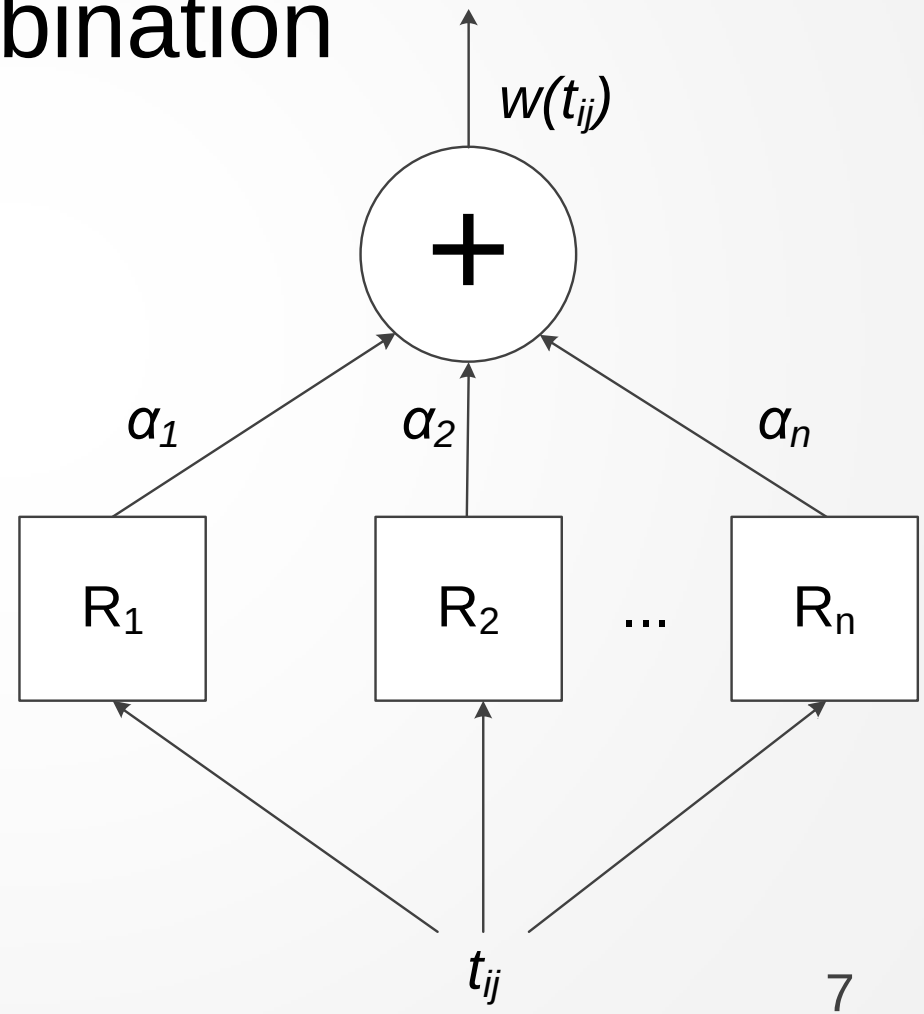
Construction of a Personalized Matrix



Construction of a Personalized Matrix

- Weights as a combination of raters
  - generic
  - personalized

$$w(t_{ij}) = \sum_k \alpha_k R_k(t_{ij})$$



Construction of a Personalized Matrix

- Generic
  - Terms frequency rater
  - Terms location rater
  - Relevant domain terms rater

$$w(t_{ij}) = w_i \quad \text{if } t_i \in S_j \cap C_d$$

$$w(t_{ij}) = 0 \quad \text{else}$$

Construction of a Personalized Matrix

- Generic
  - Terms frequency rater
  - Terms location rater
  - Relevant domain terms rater
- Personalized
  - Knowledge rater
  - Annotations rater
    - Highlights made by a particular user
    - Popular highlights

Evaluation

- Summarizer as a REST web service
- Integrated with the educational system ALEF
  - Adaptive Learning Framework
  - Learning flow
  - Collaboration/creation flow
  - Principles of Web 2.0 (commenting, tagging etc.)

ALEF

Administrácia | Debug | C | Lisp | SI | Prolog

Robert Moro (administrátor) | Odhlásiť

Filter:

Texty | Cvičenia | Otázky

- Predslov [0.0]
- Úvod - Paradigmy programovania [0.0]
  - Procedurálne programovanie [0.0]
  - Objektovo-orientované programovanie [0.0]
  - Deklaratívne programovanie [0.0]
  - Aplikatívne programovanie [0.0]
  - Programovanie ohraničeniami [0.0]
  - Vizuálne programovanie [0.0]
  - Paradigmy v súčasnosti [0.0]
- Výrazy [0.0]
- Základné prvky jazyka lisp [0.0]
- Programovacie techniky [0.0]
- Pravidlá dobrého programovania [0.0]

Ak sa pri aplikatívnom programovaní ako základný výrazový prostriedok použiju funkcie, vrátane funkcií vyšších rádov (t.j. takých, ktoré operujú nad inými funkciami), hovoríme o *funkcionálnom programovaní*. Ak sú základným výrazovým prostriedkom relácie, opísané pomocou logických predikátov, hovoríme o logickom programovaní.

Vo funkcionálnom programovaní sa výpočet opisuje výrazom.

Vo funkcionálnom programovaní sa program chápe ako množina funkcií. Na rozdiel od procedurálneho programovania, ktoré vychádza z modelu výpočtov založeného na von Neumannovej architektúre počítača, opiera sa funkcionálne programovanie o tzv. lambda počet ako jednoduchý model výpočtov. Základné pojmy funkcionálneho programovania, ako funkcia, výraz, zloženie výrazov, rekurzívna definícia funkcie sa podrobne vysvetľujú v prvej časti tejto učebnice. Takisto sa tu rozoberajú základné funkcionálne programovacie techniky, ako jednotlivé vzory rekurzívnych definícií funkcie, programovanie filtrov, generátorov, programovanie pomocou funkcií vyšších rádov (funkcionálov).

V logickom programovaní je výpočet dokazovaním dopytu.

Pri logickom programovaní sa ako programovací jazyk využíva predikátová logika. Základom je interpretácia implikácií ako deklarácií procedúr. Ide o tzv. procedurálnu interpretáciu predikátového počtu prvého rádu. Vytvoriť logický program znamená sformulovať sústavu axióm opisujúcich triedu riešených úloh. Špeciálnu úlohu, ktorú treba vyriešiť, treba sformulovať ako cieľový príkaz. Je to formula predikátového počtu, ktorá sa zapíše v špeciálnom tvare. Výpočet je potom dôkaz, že cieľový príkaz (dopyt) je logický dôsledok množiny axióm, tvoriacej program.

Základné pojmy logického programovania ako klauzula, predikát, term, odvodenie odpovede na zadaný dopyt spolu s programovacími technikami logického programovania sa podrobne vysvetľujú v druhej časti tejto učebnice.

Sumarizácia - Ohodnoť a zlepši si tak svoje skóre!

Aplikatívny program opisuje výpočet výrazom. Posun smerom k deklaratívnemu prístupu k programovaniu možno sledovať pri aplikatívnom programovaní. Jeho vyhodnotením sa získa požadovaný výsledok. Vo výraze sa neurčujú žiadne podrobnosti výpočtu ako napr. spôsob a miesto uloženia medzivýsledkov. Vo výrazoch sa teda kladie dôraz na hodnoty samotné a nie na organizáciu ich uloženia. Vo funkcionálnom programovaní sa program chápe ako množina funkcií. V logickom programovaní je výpočet dokazovaním dopytu. Pri logickom programovaní sa ako programovací jazyk využíva predikátová logika. Základom je interpretácia implikácií ako deklarácií procedúr. Špeciálnu úlohu, ktorú treba vyriešiť, treba sformulovať ako cieľový príkaz. Je to formula predikátového počtu, ktorá sa zapíše v špeciálnom tvare. Základné pojmy logického programovania ako klauzula, predikát, term, odvodenie odpovede na zadaný dopyt spolu s programovacími technikami logického programovania sa podrobne vysvetľujú v druhej časti tejto učebnice.

Slovné hodnotenie sumarizácie...

Na ďalšiu

Tvoje skóre

13.8 ?

V priebežnom hodnotení sú 34 študenti pred Tebou!

Nahlásené chyby

V tejto časti kurzu nie sú hlásené žiadne chyby.

Tagy

moje | populárne

Vo vzdelávacom obsahu taguj významné pojmy. Pomôžu tebe aj tvojim spolužiakom sa rýchlejšie orientovať v texte.

+ Pridaj tag

Externé zdroje

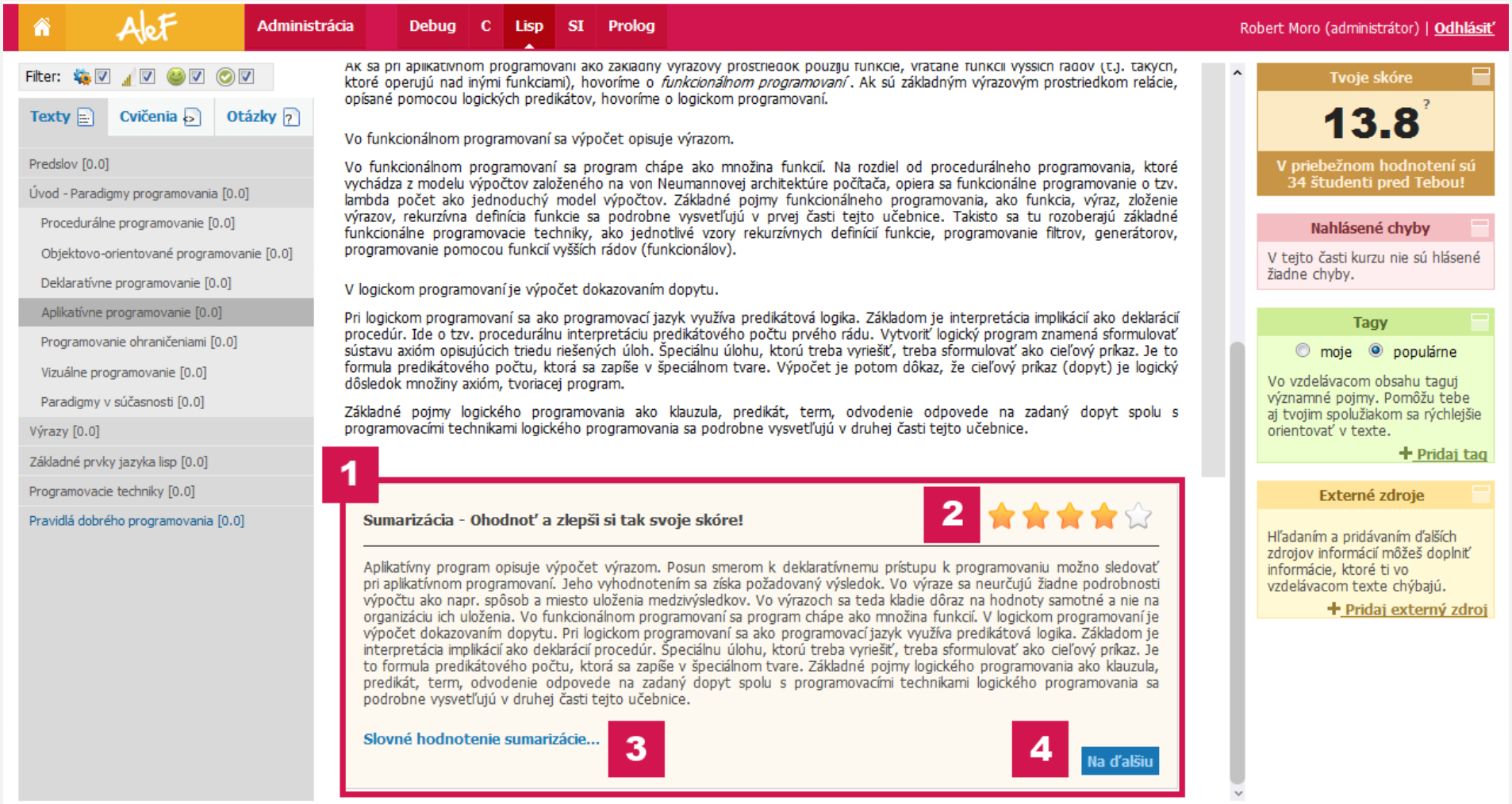
Hľadaním a pridávaním ďalších zdrojov informácií môžeš doplniť informácie, ktoré ti vo vzdelávacom texte chýbajú.

+ Pridaj externý zdroj

Evaluation

- 2 experiments
  - Functional and Logic Programming course
  - Principles of software engineering course
- Summarization considering relevant domain terms vs. generic variant
- Summarization considering annotations vs. generic variant

Summary Rating



Summary Variants Comparison

Sumarizácia - Ohodnoť a zlepši si tak svoje skóre!

1

1 V tejto kapitole vysvetlíme základné princípy logického programovania na príkladoch v programovacom jazyku prolog (z angl. programming in logic). rádu. Na základe príkladu potom v ďalšej kapitole vysvetlíme princíp logického programovania formálnejšie. Vieme už, že logické programovanie spolu s funkcionálnym programovaním sa označuje ako aplikatívne programovanie. Sústredíme sa na použitie už známych princípov (z funkcionálneho programovania) v logickom programovaní. V logickom programovaní problém špecifikujeme množinou formúl. Logické programovanie sa zakladá na postupoch, ktoré sa používajú pri dokazovaní teorém v predikátovej logike prvého rádu. V tomto systéme sa dokazuje zadaná hypotéza. Možno ich zapísať takto: A ak B a C a ... Programátor chápe takúto klauzulu ako procedúru: aby sa vyriešil problém A, redukuj ho na B a C a ... Logické programovanie je deklaratívne. V logických programoch sa nepoužívajú riadiace štruktúry (napr. cyklus while) ako ich poznáme napr. z programovacích jazykov C alebo pascal.

2 V tejto kapitole vysvetlíme základné princípy logického programovania na príkladoch v programovacom jazyku prolog (z angl. programming in logic). Vieme už, že logické programovanie spolu s funkcionálnym programovaním sa označuje ako aplikatívne programovanie. Logické programovanie sa zakladá na postupoch, ktoré sa používajú pri dokazovaní teorém v predikátovej logike prvého rádu. V logickom programovaní sa používajú tzv. Hornove klauzuly (t.j. formuly s najviac jednou kladnou zložkou). Na dôkaz zadaného cieľa sa využíva metóda rezolvencie, ktorú možno pre formuly v tvare Hornových klauzúl efektívne automatizovať. Významnú úlohu má mechanizmus unifikácie, ktorý umožňuje odovzdávanie parametrov, výber a konštruovanie údajov. Dôležité je, že programovací jazyk (prolog) poskytuje stratégiu dôkazu zadaného cieľa zadarmo – netreba ju programovať. Logické programovanie je deklaratívne. Logické programovanie je vhodné najmä na riešenie takých problémov, kde vieme určiť objekty, ktoré patria do problémového prostredia a vzťahy medzi nimi.

Porovnaj, ktorá sumarizácia je lepšia:

1 = 2 2

Slovné hodnotenie sumarizácie...

Na ďalšiu

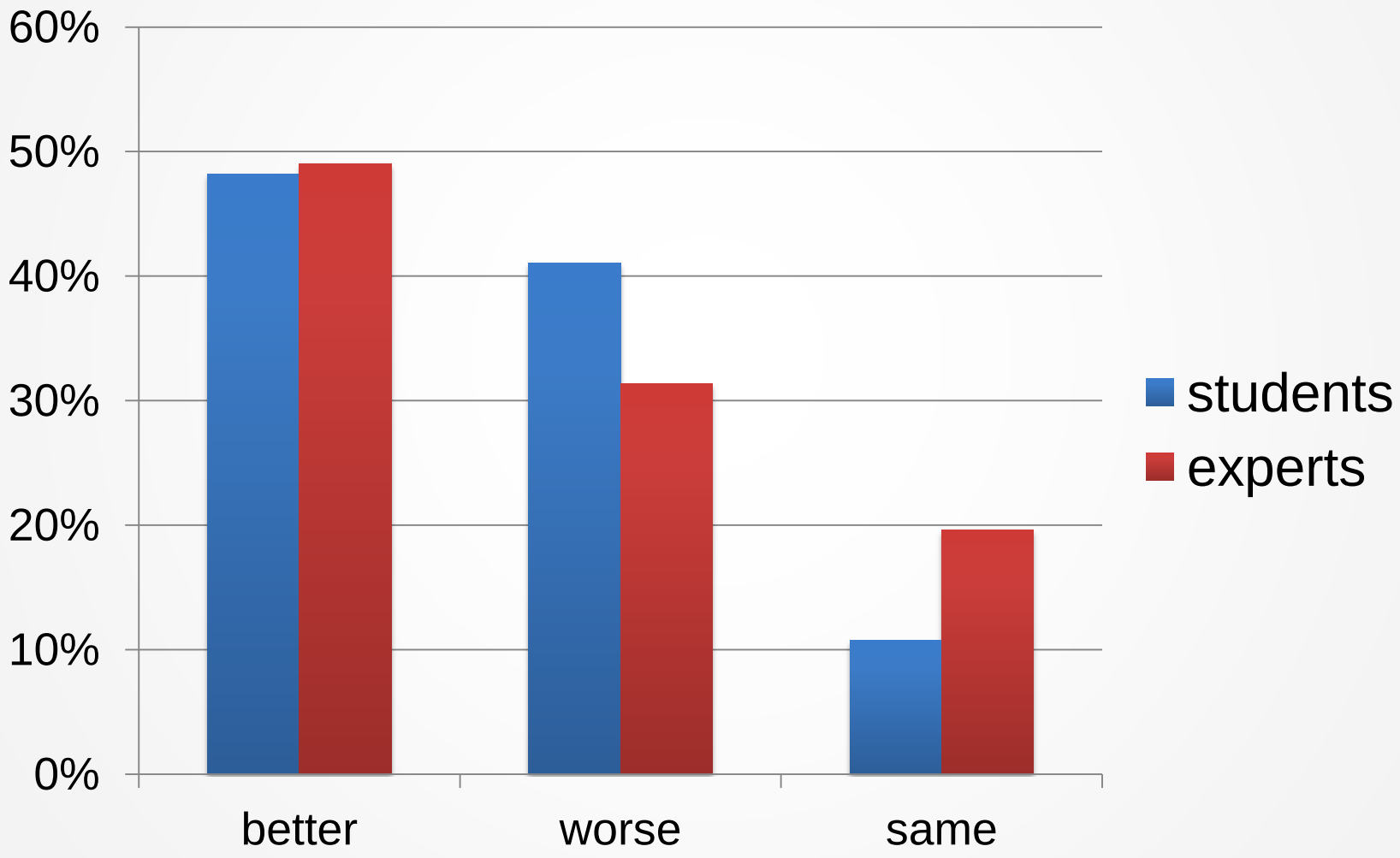
Evaluation

- 75 students, 5 experts, 303 educational texts
- 2242 summary ratings, 479 answers to the follow-up questions
- 385 summary variants comparisons by experts

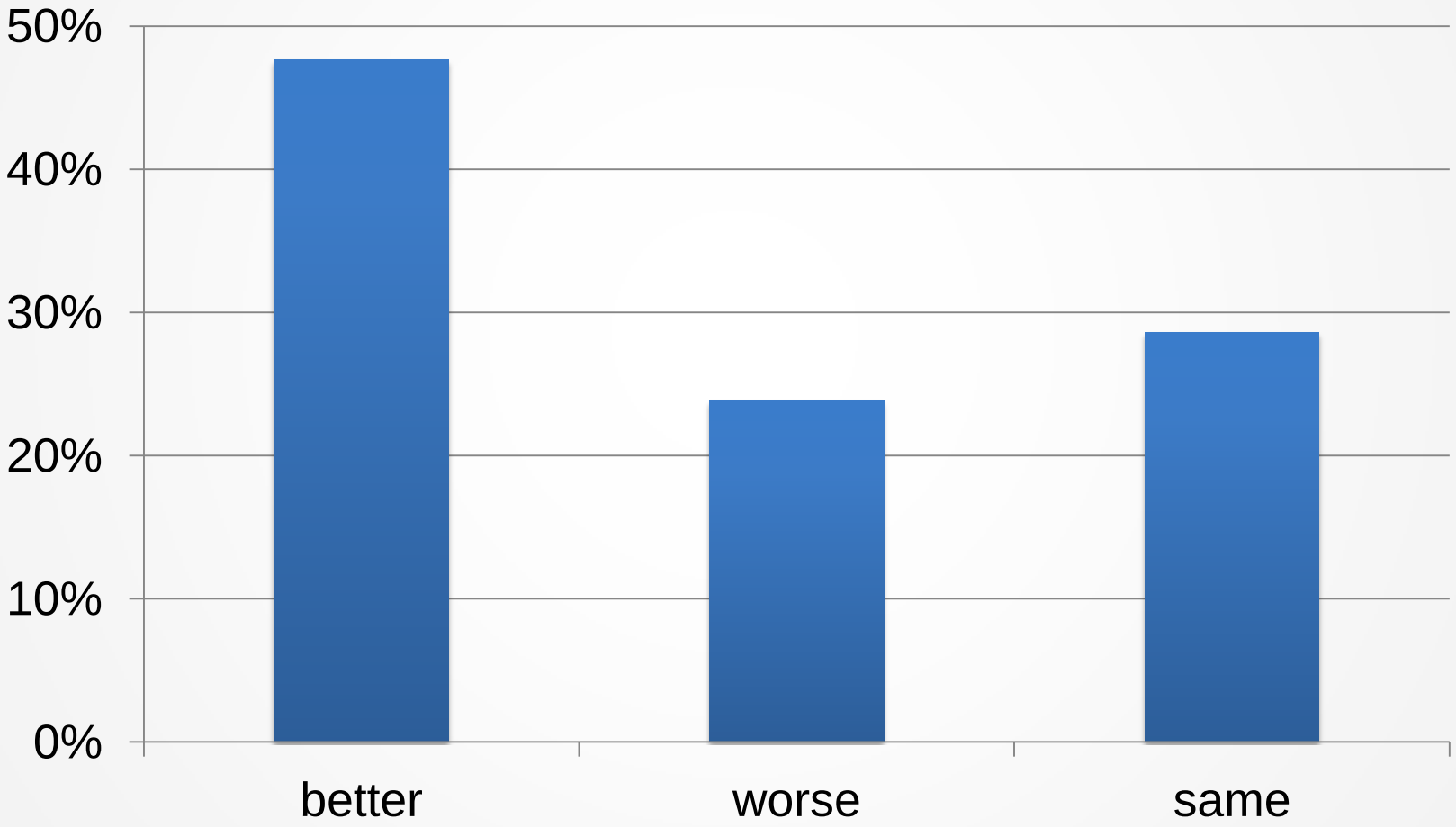
1st Experiment - Summary Ratings

	Generic	Relevant domain terms
No. of ratings	143	135
Mean	3.538	3.793
Variance (n-1)	1.518	1.419

1st Experiment



2nd Experiment



Conclusions

- Considering the *relevant domain terms*, as well as *annotations*, during the summarization process leads to better summaries in comparison to the baseline generic variant
- Summarization can be used to
  - Decide the document relevance
  - Help students to revise

Contribution

- Method of raters' combination
  - Allows considering various parameters or context of the summarization
- Specific raters that take into account
  - Terms relevant in the domain
  - Level of knowledge of a particular user
  - Annotations in the form of highlights

Future Work

- Automatic (dynamic) setting of the raters' combination parameters based on
  - Reliability of the raters' sources of information
  - Category of the summarized document
- Using user tags instead of relevant terms identified by experts