SuggestBot: Using Intelligent Task Routing to Help People Find Work in Wikipedia

Dan Cosley, Dan Frankowski, Loren Terveen, John Riedl

presented by: Gianluca Demartini

July 6, 2007

Outline

1. Introduction
2. System
3. Recommendation Algorithms
4. Experiments
5. Conclusions

Scenario

- Wikipedia lives thank to its contributors
- It is made of community-maintained artefacts of lasting value (CALV)
- Similar communities are iMDB, slashdot.org

Wikipedia

- A full dump (with history of pages) was 700GB in 2006
- Reasons for participating are similar to open source: learning, status, belonging
- Bots: automated or semi-automated editing of pages
- People tag articles they think need work

Motivation

- Member-maintained communities need contributions
- Reducing the *cost of contribution* increase motivation
- Goal: make it **easy to find work** to do
  - interesting
  - that need work

Intelligent task routing definition

Intelligent task routing

- reduces the cost of finding work
- **matches people with tasks** they are likely to care about

as a mechanism for increasing contribution

Intelligent task routing on Wikipedia

With Intelligent task routing using

- history of edits
- text matching
- link following
- collaborative filtering

people edit four time more often.

Outline

1. Introduction
2. System
3. Recommendation Algorithms
4. Experiments
5. Conclusions

SuggestBot

- SuggestBot provides recommendations only **on request**
- SuggestBot edits the **user talk pages** adding recommendations

Architecture

Four steps:

- Pre-processing Wikipedia
- Modelling user's interests
- Finding candidate articles
- Make recommendations

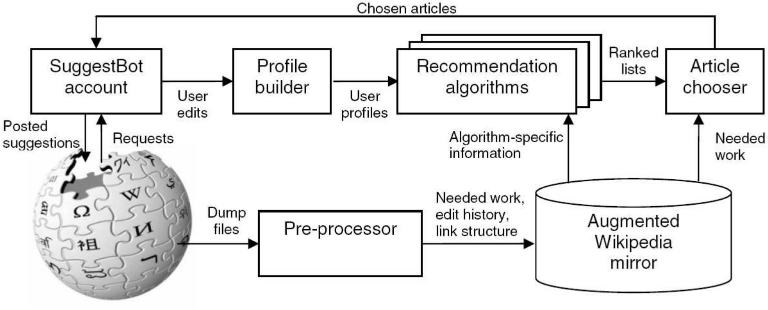

Figure: SuggestBot architecture

Pre-processing

SuggestBot processes the wikipedia looking for users' annotation.

SuggestBot considers 6 types of work

- Stub: short article needs more info
- Cleanup: need rewriting
- Merge: articles need to be combined
- Source: need citation
- Wikify: the text is not in the correct style
- Expand: long article needs more info

Modelling interests

The User's Interests profile

- is build implicitly (no user participation)
- is the set of **article titles** that have been *edited* by the user
- does not contain more than 500 articles
- does not consider "vandalism reversion" edits
- considers multiple edits as single edit

Outline

1. Introduction
2. System
3. Recommendation Algorithms
4. Experiments
5. Conclusions

Finding candidate articles

SuggestBot does not recommend already edited articles
SuggestBot finds related articles based on

- similarity of text: user's profile as query against full text
- connections through links
- connections through co-editing

Connections through links

- SuggestBot uses links created by the users (citation network)
- SuggestBot ignores date-related links
- ALGO:

Initialize items in the profile to have a score of 1

{Expand profile until we have enough articles}
while $depth < MaxD$ and ($|i|$ with $i.score > 0) < N$ **do**
  for all links to items l from items with $i.score > 0$ **do**
    $l.score \leftarrow l.score + 1$
  end for
  $depth \leftarrow depth + 1$
end while

Remove items from original profile

{Penalize items with many or few links.}
for all items i with $i.score > 0$ **do**
  $L \leftarrow number\ of\ links\ to\ i\ in\ Wikipedia$
  $i.score \leftarrow i.score/log(count\ of\ articles/abs(BestL - L))$
end for

Penalization function:

$$score := \frac{score}{log(\frac{\#art}{|BestL - L|})} \tag{1}$$

Connections through co-editing

- Find people whose history is similar using Jaccard metric for similarity between profiles
- Give credit to items based on the users' similarity

Connections through co-editing

{Find all my neighbors}
for all users u who have edited any item $i \in T$ **do**
  $U \leftarrow$ all items edited by u
  $J \leftarrow \frac{|T \cap U|}{|T \cup U|}$ {Jaccard similarity with this neighbor}
  {only recommend if similar enough}
  if $J > MinJ$ **then**
    for all items $i \in U$ **do**
      $i.score \leftarrow i.score + J$ {weighted credit}
      $i.count \leftarrow i.count + 1$
    end for
  end if
end for

Remove items edited by few others, or edited by the user t

Filtering results

- Both algorithms tend to recommend popular or controversial articles
- SuggestBot drops the top 1% of the most edited articles

Outline

1 Introduction

2 System

3 Recommendation Algorithms

4 Experiments

5 Conclusions

Setup

- In 6 months 1200 people got recommendations
- Comparison based on number of edited recommendations
- Three different experiments:
  - compared to random suggestions: 4 times more edited
  - comparing text-similarity, links, co-edit.
    Similar performances with differences:
    - text reco: focuses on rare word
    - links:biased by categories (link circles)
    - co-edit: often edited articles
    - Using meta-search techniques to combine results would help
  - removing noise: not considering *minor* edits does not improve

Recommender	Edited	Total	Percent
Co-edit	29	726	4.0%
Text	34	790	4.3%
Links	25	742	3.4%
Random	8	836	1.0%
Total	96	3,094	3.1%

Table 2: Suggestions edited within two weeks of posting. Personalizing recommendations improves performance compared to random ($\chi^2(2, 3094) = 16.77, p < 0.01$).

Outline

1 Introduction
2 System
3 Recommendation Algorithms
4 Experiments
5 Conclusions

Conclusions

Simple algorithms gave strong results

Intelligent task routing

- can dramatically increase members' contributions
- is most useful where the tasks are numerous and heterogeneous

Future steps:

- incorporate meta-search techniques
- remove noise
- give to the user the possibility to edit the profile with low cost

The End

Q&A