Identifying experts through a framework for knowledge extraction from public online sources

Simon Buelens and Mattias Putman

Supervisors: Prof. Dr. Ir. Filip De Turck, Dr. Ir. Elena Tsiporkova, Dr. Ir. Tom Tourwé, Ir. Anna Hristoskova, Ir. Tim Wauters

Abstract—

Researchers are loosing too much valuable time searching for related research material. This article

Keywords—**author disambiguation, data processing, clustering**

I. Introduction

Researchers loose valuable time searching for research material related to their field of expertise. The process of finding and verifying experts is extensive and troublesome. The aim of this article is creating a framework that is capable of retrieving publications and related information from online sources, analyzing this information and linking it to the correct author and enabling users to search for experts for a given subject. The main focus is on the disambiguation of authors, the classification into clusters and the extensibility of the framework.

II. Theoretical Model

The foundation of the framework is based on the five following observations:

1. All instances are different authors until proven otherwise.
2. No decision is made permanent.
3. Any information is considered partial information.
4. A constantly changing input asks for a constantly changing output.
5. The stream of information is endless.

Starting from this foundation, the article lays out the rudiments of the framework, starting with a theoretical model. This model is composed of three layers, combining the structural, informational and algorithmic aspects that emerge from dealing with the difficulties related to author disambiguation. This is achieved by a graph representation where the authors are phased in three different levels. At the highest level is the family name, below are authors that are considered unique (a cluster) containing instances of names that are linked to the publications. This allows name-matching and regrouping without losing information.

The theoretical model also contains a summary of the different rules. They drive the entire flow of the framework by converting new information into similarities between instances. The four rules that are examined are:

- *Community Rule* Exploiting the fact that authors often work together with the same co-author. **??** gives a visual representation of how this works.
- *Interest Rule* The subjects of publications of the same author are usually located within the same field of research.
- *Email Rule* Authors with the same email address, are most likely the same person.
- *Affiliation Rule* Authors are more likely to work at one affiliation at a given time.

III. Pipes and Filters

The core of the implementation of the framework is the simplicity achieved by pipes and filters. They allow for modifiability and extensibility as pipes processing new sources or calculating new similarities between instances can easily be added to the system. An overview of the different pipes and the flow that runs between them is shown on Figure 2. The expressiveness of StatefulPipe and they scalability of AsyncConnection in combination with a shared key-value store enables out-of-the-box and carefree distribution.

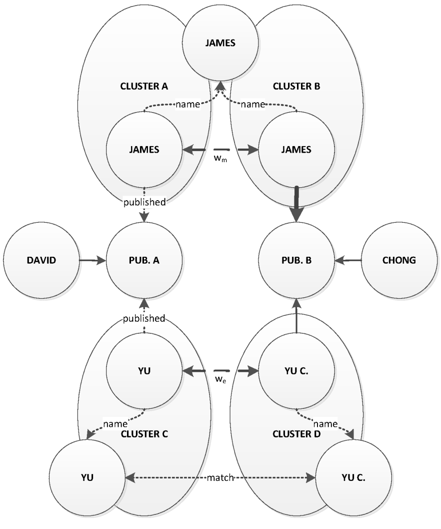

Fig. 1

The co-author rule in action: comparing the two instance of James, a similarity (w_m) is added as the co-authors Yu and Yu C. match.

IV. Clustering

The article focuses a lot on the clustering process. This process is responsible for calculating what instances match to same author, based on similarities between the instances. Everytime a rule computes a new similarity, it is possible reclustering has to occur. As there is a constant influx of information, there is a need for a dynamic approach that maintains the cluster quality.

The article gives an in-depth explanation of the dynamic clustering algorithm described in [**?**]. The algorithm itself depends on the minimum cut tree algorithm. The sequential Gusfield's algorithm described

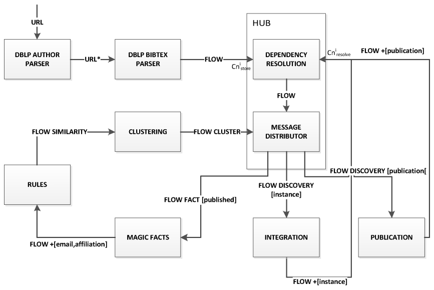

Fig. 2

An overview of the different pipes and the flow running between them.

in [?] is implemented.

The article also proposes an addition to the original clustering algorithm. It introduces a new case, acting the same as case 1, but being executed instead of case 3 on the occasion that $2 * \frac{cutvalue(C_v, C_u)}{|V|} <= \alpha/2$. This anticipates the fact that a lot of similarities with lower weights occur and they often take place in succession. Often this results in case 3 being calculated multiple times in succession, before finally merging the clusters together. By introducing this new case, the resource-intensive execution of case 3 is switched for the very quick execution of case 1.

The clustering process itself is also a pipe and as noted before, in section III, the shared key-value store Redis allows this process to be distributed without any trouble.

V. Results

Fig. 3

Performance test of the matchmaker on Laptop (2GHz processor, 1GB RAM) and ALIX (500MHz processor, 256MB RAM)

Fig. 4

Test of influence of parameters on performance on Laptop (2GHz processor, 1GB RAM)

VI. Conclusions and future work

References

[1] Donoho A, Costa-requena J, Mcgee T, Messer A, Fiddian-green A, Fuller J. *UPnP Device Architecture 1.1.* Oct. 2008.

[2] Bauer C. *Cling UPnP.* 22 Mar. 2011. Available from: http://teleal.org/projects/cling/

[3] Smith M, Welty C, McGuinness D. *OWL - Web Ontology Language.* 5 Apr. 2011. Available from: http://www.w3.org/TR/owl-guide/

[4] Martin D, Burstein M, Hobbs J, Lassila O, McDermott D, McIlraith S, et al. *OWL-S - Semantic Markup for Web Services.* 21 Sep. 2010. Available from: http://www.w3.org/Submission/OWL-S/

[5] SemWebCentral. *OWL-S Service Retrieval Test Collection: Project Info.* May 16 May. 2011. Available from: http://www.semwebcentral.org/projects/owls-tc/