A Graph-based Disambiguation Approach for Construction of an Expert Repository from Public Online Sources

Anna Hristoskova[1], Elena Tsiporkova[2], Tom Tourwé[2], Simon Buelens[1], Mattias Putman[1] and Filip De Turck[1]

[1]*Department of Information Technology, Ghent University, G. Crommenlaan 8 (Bus 201), B-9050 Ghent, Belgium*

[2]*Sirris, Software Engineering & ICT Group, A. Reyerslaan 80, 1030 Brussel, Belgium*

Keywords: Author Disambiguation, Expert Finding, Clustering, Data Processing, Graph Data Model.

Abstract: The paper describes a dynamic framework for the construction and maintenance of an expert-finding repository through the continuous gathering and processing of online information. An initial set of online sources, relevant to the topic of interest, is identified to perform an initial collection of author profiles and publications. The extracted information is used as a seed to further enrich the expert profiles by considering other, potentially complementary, online data sources. The resulting expert repository is represented as a graph, where related author profiles are dynamically clustered together via a complex author disambiguation process leading to continuous merging and splitting of author nodes. Several rules are developed that assign weights to the links in the graph based on author similarities such as name, affiliation, e-mail, co-authors, and interests. Dynamic clustering of the authors depending on these weights results in the identification of unique experts for a specific domain.

The developed disambiguation and author clustering algorithms are validated on several authors with varying name notations showing an improvement on the identification of unique profiles of 28% compared to the results from DBLP.

1 INTRODUCTION

Nowadays organizations are looking for opportunities to adopt and implement a formal, structured and focused approach for the identification of individuals having a particular expertise and background. However, there are currently no adequate tools available to support such needs. Although a large pool of data is available on the web, it is often gathered manually, which is a time intensive, tedious and error-prone process, due to the fact that the data is not centralized, is available in different formats, can be outdated or contradictory. Leading search engines mainly provide keyword-based results in response of a search query. This is limited in terms of accuracy and efficiency of information comprehension as one still has to manually search for more data on experts, their level of expertise and their connections.

Research on identifying experts from online data sources has been gradually gaining interest in the recent years. In (Balog and de Rijke, 2007) similar experts are found based on an initial collection of example experts. The authors define several ways of representing experts: through their collaborations, the documents they are associated with, and the terms they are associated with. Furthermore from the comparison on the number of the initial set of experts they conclude that larger input samples lead to higher similarity scores. Similar to this the authors in (Zhang et al., 2010) focus on utilizing personal and relationship information for finding experts in a social network. In a first phase, the person's local information is used to estimate an initial expert score and select the top ranked people as candidates. In the second phase, a propagation-based process propagates one's expert score to the people with whom he/she has relationships. Another research challenge is choosing the right communication channel that is commonly used by the authors, to query for specific experts. (Stankovic et al., 2011) propose just such an approach for adapting the expert search process to the given topic of expertise, by relying on Linked Data metrics.

Other approaches use semantic technologies such as OntoFrame (Jung et al., 2007a) to identify experts

Hristoskova A., Tsiporkova E., Tourwé T., Buelens S., Putman M. and De Turck F..
A Graph-based Disambiguation Approach for Construction of an Expert Repository from Public Online Sources.
DOI: 10.5220/0004192300240033
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (ICAART-2013), pages 24-33
ISBN: 978-989-8565-39-6

from CiteSeer and full text documents. Firstly, author information on experts with the same co-authors is merged. Next, topically-classified papers are constructed through topic extraction based on full text analysis. A last step is the use of SPARQL queries to retrieve URI-based 'Persons by Topic' from an RDF triple store.

Adopting the strengths of the Semantic Web is a sufficient first step in finding experts for a specific domain. A necessary requirement is the actual identification of unique author profiles based on the extracted personal and community data. This demands for more formal ways of representing author profiles and the development of complex disambiguation methods able to differentiate between distinct authors.

The presented paper supports this upcoming trend by creating a framework that constructs an expert-finding repository in an incremental fashion through the continuous gathering and processing of user-related information from a variety of online sources. This allows users to query the expert repository with a set of keywords defining the subject area they want to investigate. The outcome is a list of authors, ranked by decreasing level of expertise on the specific subject. Each author is accompanied by a profile, containing a list of papers, highly touted co-authors and any other information the user might find useful. The main novelties are

1. a graph model representing author profiles, consisting of information such as e-mail, affiliation, publications, interests, links to co-authors, etc;
2. a complex disambiguation process based on the dynamic clustering of the related author profiles that continuously splits and merges author nodes.

This disambiguation algorithm compares the gathered author information represented in the graph in order to assign similarity weights between related authors. These weights are used to calculate the probability that several authors are equal resulting in the identification of unique author profiles. The developed disambiguation and author clustering algorithms are validated on several authors with varying name notations showing an improvement on the identification of unique profiles of 28% compared to the results from DBLP.

The paper starts with an overview in Section 2 of the related work on expert-finding applications. Its outcome is the identification of several research challenges to be tackled in Section 3. The actual implementation is thoroughly explained in Section 4, which makes use of a graph representation for the expert model. The clustering and disambiguation process, responsible for identifying unique authors, are the key components. The article concludes with a comparative analysis of the results from this approach and DBLP in Section 5. Finally, the main conclusions and future improvements are drawn in Section 6.

2 RELATED WORK

Several works on the topic focus on the adoption of advanced information retrieval, machine learning and reasoning techniques, while considering applications such as research collaboration or enterprise expert search.

(Hofmann et al., 2010) explore the integration of contextual factors that have been found to influence human expert-finding, into content-based approaches for finding similar experts. Such factors include accessibility, reliability, physical proximity, and up-to-dateness. A similarity score between experts is assigned based on the retrieved expert profile content and the contextual factors playing a role such as working for the same group.

Probabilistic methods such as (Fang and Zhai, 2007) develop a framework for studying expert-finding. Specifically, candidates are ranked according to the probability that a candidate is ”relevant” to the topic (i.e., expertise) specified in a query using two models, i.e., candidate generation models and topic generation models. Topics are mapped to candidate experts based on profile and document retrieval. In a similar fashion the process of expert search in (Balog et al., 2009) is based on two probabilistic models. The first model uses the associations between people and documents to build a candidate model and match the topic against this model, and the second matches the topic against the documents and then uses the associations to amass evidence for a candidate's expertise.

These traditional methods usually estimate the relevance between the query and the support documents of candidate experts using a language model. However, this approach lacks the ability of identifying semantic knowledge, resulting in experts that cannot be found due to no occurrence of the query terms in the supporting documents. This is tackled in (Zhang et al., 2008) where people with expertise knowledge on a given topic are identified using a mixture model combining probabilistic techniques to capture the semantic relevance between the query and the support documents of candidate experts. The model identifies additional topic themes that relate the query terms to support documents. The Rule Responder in (Boley and Paschke, 2007) reuses knowledge (ontologies and rules) to query data from Web resources such as vCards, FOAF or SIOC profiles. It represents a

Web-based communication infrastructure for expert querying and redirection to other services able to retrieve unanswered queries. Rule-based reasoning is combined with case-based reasoning in (Tung et al., 2010) to retrieve experts able to solve a specific system problem. The rule-based reasoning adopts experts' knowledge in a diagnosis phase by categorizing similar problem types in advance narrowing down the searching spaces. The case-based reasoning locates a problem according to the classified categories in a retrieval phase. Finally, based on domain knowledge of personal expert profiles, the appropriate expert is identified able to solve the problem.

Ontology-based inference is also adopted to support collaboration between researchers. The Semantic Campus by (Sriharee and Punnarut, 2007) proposes a construction of a Semantic Web application that represents the social network of the academics in a university (i.e. King Mongkut's Institute of Technology North Bangkok). Extracted data available on the web site of the university is analyzed with respect to its association to terms defined in an ontology and between people in the university to reveal links between academics. The semantic platform in (Jung et al., 2007b) focuses on the verification and the tracing of information using an information dissemination platform and Semantic Web-based services. Services include information dissemination supporting reliable information exchange among researchers (including keyword-based document search and related document search with version tracing) and knowledge service providing unrevealed information. (Pavlov and Ichise, 2007) propose a method for building link predictors in networks, where nodes represent researchers and links - collaborations. The method extracts structural attributes from the graph of past co-authorships and uses them to train a set of predictors using supervised learning algorithms. These predictors can then be used to predict future links between existing nodes in the graph and thus suggest future collaborations.

Some other recent contributions in the area of integrating web data consider a set of challenging research topics such as data cleansing, profiling, and entity resolution. ProLOD in (Böhm et al., 2010) is an iterative and interactive methodology for profiling linked open data. The process divides data into groups using K-means clustering. It determines the actual schema on a group-level by finding equivalent attributes. Afterwards it rethinks grouping decisions in order to revise them for refining the profiling result. The algorithm in (Pu et al., 2010) annotates unbounded text streams with entities of a structured database. This allows one to correlate unstructured and dirty text streams from sources such as e-mails, chats and blogs, to entities stored in structured databases. Relations among entities are currently not considered.

In terms of scale, the problem of finding experts is usually tackled by mining vast online data sources such as Wikipedia and CiteSeer (see (Jung et al., 2007a; Whitelaw et al., 2008)), in order to gather a sufficiently large data repository containing information on persons, articles, social links. The advantage is achieving high coverage of the experts, active in a certain domain, and relatively complete expert profiles in space and time. Unfortunately, such large data collections contain a substantial proportion of noisy data (contradictions, duplicates) and the achieved degree of accuracy cannot be estimated in a reliable way.

The proposed approach in this paper focuses on the identification of the main research challenges to be considered in order to determine unique expert profiles. An expert repository is constructed in an incremental fashion using an initial set of online data sources enriched with extracted information from the author profiles. The approach proposes a graph as a representation model for the resulting expert profiles. Related author profiles are dynamically clustered together into unique authors via a complex author disambiguation process leading to continuous merging and splitting of graph nodes. Authors are identified based on several different metrics: name similarity, affiliation, co-authors, e-mail and interests.

3 RESEARCH CHALLENGES

Most applications that automatically gather user information from the web serve personalization or recommendation purposes. However, there are many other potential applications, as for instance the identification of experts in a particular technological domain for the purpose of technology scouting, partner matching for research proposals, or visualization of research activities and experts within geographical regions *e.g.* in the context of technology brokerage. Going beyond recommendation and personalization applications often imposes more strict requirements:

1. Very *high* (if not complete) *coverage* over the domain should be attained. This requires information extraction from multiple heterogeneous data sources; structured (LinkedIn, Twitter), semi-structured (ACM DL, DBLP) and unstructured (conference pages).
2. The data needs to be *up-to-date* at all times resulting in a data streaming pipeline that continuously

presents newly gathered information to approve new, or revoke previously taken decisions.

3. *High accuracy/reliability* should be guaranteed. This demands the development of advanced disambiguation techniques and the quantification of the different sources in terms of reliability and trustworthiness (e.g. distinguish between doubtful and reputable sources).
4. It should be possible to *rank the experts* in terms of impact and relevance. Identification of adequate criteria and metrics, which most probably will be application- and problem-dependent, allowing to perform multi-criteria decision analysis is necessary.

These requirements are taken into account in the next section using a bottom-up approach to building the expert-finding repository.

4 BOTTOM-UP CONSTRUCTION OF AN EXPERT-FINDING REPOSITORY

We propose a bottom-up expert-finding approach, which implements an entity resolution method allowing for reliable disambiguation of authors of scientific articles. Its internal functioning is split up in three main components as presented in Figure 1: *gathering data* from various online sources (publications, author profiles, online presentations), improving accuracy through *data cleaning* and *disambiguation* between authors, and *analyzing* and clustering this data into unique author profiles. These steps demand additional requirements such as customized information extraction techniques from the identified sources and a formal way of representing the author profiles supporting the disambiguation and clustering process (Section 4.1).

The result from the *data gathering* step is high coverage through the incremental extension of an initial set of online sources targeting the application domain in question. These sources serve as seeds for the bootstrapping and the incremental growth of the repository using web scraping techniques to extract information such as an initial list of authors, article title, abstract, co-authors, and affiliation. Subsequently, using the extracted information additional sources are considered, such as Google Scholar, DBLP, Microsoft Academic Search, in order to search for authors and co-authors and identify additional published material. This leads to a broader set of actors in the field, technology-related publications, research activities and author career evolution. The expert repository is constantly dynamically updated with this infinite stream of information.

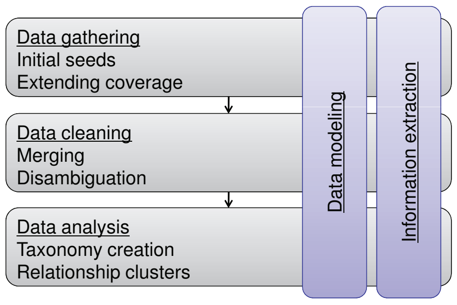

Figure 1: A bottom-up approach to building an expert-finding repository.

The collected data consists of partial information on entities (authors) and relationships (links between authors), which are often inconsistent and conflicting. For instance, an author's name is not a unique reference to a person, as there might be multiple authors with the same name or the name can be spelled out differently or change throughout time. It should be possible to discriminate between different individuals with similar names. *Merging* and *disambiguation* are required to guarantee that an expert profile and associated publications refer to a unique author. During the *data collection* phase each new piece of author information linked to an author's name is stored as a new entity in the repository, even if that name is already present. This is necessary in case the same name is connected to different authors. The *disambiguation* of authors consists of a number of rules (detailed in Section 4.2.2) which inspect several entities in the repository and define the probabilities that names, typically connected to a publication or a profile, represent a unique author. The *merging* step clusters the names so they would reference the same author using the probabilities calculated during the disambiguation phase (Section 4.2.1).

The next sections focus on the main novelty of the paper consisting of the development of a graph model for representing author profiles, disambiguation rules assigning similarity weights to author links and a dynamic minimum-cut clustering algorithm identifying unique experts based on these weights.

4.1 Graphs as a Flexible Data Model

Given that new information is constantly gathered, the results of the *disambiguation* and *merging* phases are not permanent as decisions might need to be revoked. This requires a data model enabling flexible management of the continuous stream of partial information.

The extracted information comprises of entities (authors) and relations between them, which can be easily modelled as an ontological structure. This allows representing and reasoning on the information in a general way independent of the specific domain at hand. Thus, the selected representation method is a graph-based model, as illustrated in Figure 2. This graph model is composed of three layers, combining the *structural*, *informational* and *algorithmic* aspects that emerge from dealing with the complexities related to author disambiguation.

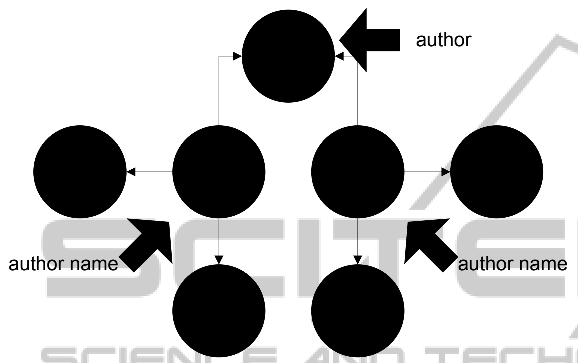

Figure 2: The extracted information is represented as a collection of nodes and edges describing (partial) information about an author ”instance”.

The *structural layer* defines the actual graph structure, which reflects the author disambiguation decisions through a change in structure. The extracted information is represented as an ”instance” consisting of a collection of nodes and edges that describe (partial) information on an author such as name, e-mail, publications, co-authors. Constructing a complete author profile amounts to finding an optimal partitioning (clustering) of instances resulting in each instance-group (cluster) representing a unique author.

The *information layer* comprises of the data itself and structures the partial author information. The authors are considered unique containing name instances linked to their publications. There is no limit on the amount of data. Every new addition to the author profile (publications, locations, events) is used to produce edge similarities in the *similarity layer* between separate authors increasing the precision of the framework. This data flow is constantly updated.

Finally, the *similarity layer* defines similarities between author instances, performs clustering and links the instances referring to the same author. This layer is responsible for the change in graph structure in the *structural layer*. Every time a new similarity is computed, there is a possibility that re-clustering will be invoked. The constant influx of information from the *information layer* requires a dynamic approach, detailed in the next section that maintains the cluster quality.

4.2 Continuous Incremental Clustering

This section describes the developed clustering algorithm, adopted by the *similarity layer* in the previous section, identifying unique experts based on the similarities between names, affiliation, e-mail, co-authors, and interests. The process is split up in two parts: a domain-independent minimum-cut tree algorithm responsible for the clustering process and the definition of domain-dependent rules assigning similarity weights to authors based on the available profile information.

4.2.1 Domain-independent Dynamic Minimum-cut Tree

Minimum-cut Tree. Similarity edges are added by the domain-dependent rules in Section 4.2.2 between author's name, e-mail address, affiliation nodes, co-authors, and author interests. A domain-independent dynamic minimum-cut tree algorithm described in (Saha and Mitra, 2006) computes clusters based on these similarity edges. Only a part of the minimum-cut tree is built as the number of authors impacted by new data entries is limited and the tree is computed over subset of nodes affecting small number of clusters. This solution guarantees efficiency while maintaining an identical cluster quality as the static version of the algorithm. The sequential Gusfield's algorithm described in (Flake et al., 2004) is implemented which calculates the minimum-cut tree of any given graph. The pseudo code is given by Algorithm 1 where the numbers represent nodes or vertices and can be chosen randomly.

$G = (V,E,w)$ denotes a weighted undirected graph with $n = |V|$ nodes or vertices and $m = |E|$ links or edges. Each edge $e = (u,v), u,v \in V$ has an associated weight $w(u,v) > 0$. Let s and t be two nodes in $G(V,E)$, the source and destination. The minimum-cut of G with respect to s and t is a partition of V, denoted as S and V/S. These partitions should be such that $s \in S, t \in V/S$ and the total weight of the edges linking nodes between the two partitions is minimum. The sum of these weights is called the cut-value and is denoted as $c(S,V/S)$.

The minimum-cut tree is a tree on V such that inspecting the path between s and t in the tree, the minimum-cut of G with respect to s and t can be obtained. Removal of the minimum weight edge in the path yields the two partitions and the weight of the corresponding edge gives the cut-value.

Gusfield's algorithm consists of $n-1$ iterations of a Maximum Flow algorithm and for every iteration a different vertex is chosen as source. The destination vertex is determined by previous iterations and is

Algorithm 1: Sequential Gusfield's Algorithm.

```
Input: $G = (V, E, w)$
Output: $T = (V, E, f)$, where T is a cut tree of G
$V(T) \leftarrow V(G); E(T) \leftarrow \emptyset$
for $tree_i, flow_i, 1 \le i \le N$ do
    $tree_i \leftarrow 1; flow_i \leftarrow 0$
end for
// $n-1$ maximum flow iterations
for $s \leftarrow 2$ to $N$ do
    $flow_s \leftarrow$MaxFlow$(s, tree_s)$
    // adjust the $tree$ with Cut$(s, tree_s)$
    // $c_1$ contains s and connected nodes, $c_2$ contains
    $tree_s$ and connected nodes
    for $t \leftarrow 1$ to $N$ do
        if $t == s \lor t == tree_s$ then
            next
        else if $t \in c_1 \land s \in c_2$ then
            $tree_t \leftarrow s$
        else if $t \in c_2 \land s \in c_1$ then
            $tree_t \leftarrow tree_s$
        end if
    end for
end for
// Generate T
for $s \leftarrow 1$ to $N$ do
    $E(T) \leftarrow E(T) \cup s, tree_s$
    $f(s, tree_s) \leftarrow flow_s$
end for
return T
```

is saved in the tree. Initially all vertices of the output tree point to node 1, but this can be adjusted after each iteration. This adjustment depends on the minimum-cut between the current source and destination. All the nodes are split in two collections, using this minimum-cut. The parent of each node is adjusted if it is on another side as its current parent, which is stored in the tree.

Adjacency Matrix. The minimum-cut algorithm described by (Saha and Mitra, 2006) identifies the adjacency matrix A of G as an $n \times n$ matrix in which $A(i, j) = w(i, j)$ if $(i, j) \in E$, else $A(i, j) = 0$. The algorithm also maintains two new variables for every vertex, the *In Cluster Weight (ICW)* and the *Out Cluster Weight (OCW)*. If $C_1, C_2, ...C_s$ are the clusters of $G(V, E)$ then *ICW* and *OCW* are defined as follows:

Definition 1. ***In Cluster Weight (ICW)*** *of a vertex $v \in V$ is defined as the total weight of the edges linking the vertex v to all the vertices which belong to the same cluster as v. That is, if $v \in C_i$, $0 \le i \le s$ then $ICW(v) = \sum_{u \in C_i} w(v, u)$*

Definition 2. ***Out Cluster Weight (OCW)*** *of a vertex $v \in V$ is defined as the total weight of the edges linking the vertex v to all the vertices which do not belong to the same cluster as v. That is, if $v \in C_i$, $0 \le i \le s$ then $OCW(v) = \sum_{u \in C_j, j \ne i} w(v, u)$*

Based on this adjacency matrix and the corresponding vertex weights *ICW* and *OCW* new edges between similar author nodes are added to the graph resulting in possible re-clustering of the entire graph structure.

Edge Addition. A similarity between two instances is represented as a new edge between two author nodes with a given weight. Using this weight, the probability that two instances belong to the same author can be derived. The weight is calculated by the disambiguation rules. There are two different possibilities that are treated separately: *inter-* and *intra-cluster* edge addition.

Intra-cluster edge addition means that both the nodes of the added edge belong to the same cluster. The result is that the cluster becomes stronger connected. A single requirements is updating the *ICW* and the adjacency matrix *A* with the new edge weight while the nodes remain unchanged.

Addition of an edge whose end nodes belong to different clusters (*inter-cluster* edge addition) is more challenging as it increases the connectivity across different clusters. This means that the cluster quality of the clusters involved is lowered and as a result reclustering might be necessary when the quality is no longer maintained. There are three identifiable cases:

1. **CASE 1:** The addition of the edge does not break the clusters involved.
2. **CASE 2:** The addition of the edge causes the clusters to be so well connected that they are merged into one. In Algorithm 2 two clusters C_u and C_v are merged into one new cluster containing the nodes of the two original clusters. This causes the *ICW* of all the nodes involved to increase and the *OCW* to decrease as all the nodes are now more connected.
3. **CASE 3:** The new edge deteriorates the cluster quality and the nodes in both the clusters have to be reclustered. All the nodes outside the set of clusters S in Algorithm 3 are replaced by a single new node x. Self loops created during this process are removed and parallel edges are replaced by a single edge with weight equal to the sum of the parallel edges. The reason we consider the clusters outside S is because S will generally be small while the other clusters will contain a lot of nodes.

In each of these three cases, the addition of a new edge results in updating the adjacency matrix A and the *OCW* of the nodes involved with the edge weight.

Algorithm 2: Merging of clusters C_u and C_v.

```
Input: $C_u$ and $C_v$
Output: Merged cluster
$D \leftarrow C_u \cup C_v$
for $\forall u \in C_u$ do
    $ICW(u) \leftarrow ICW(u) + \sum_{v \in C_v} w(u,v)$
    $OCW(u) \leftarrow OCW(u) - \sum_{v \in C_v} w(u,v)$
end for
for $\forall v \in C_v$ do
    $ICW(v) \leftarrow ICW(v) + \sum_{u \in C_u} w(v,u)$
    $OCW(v) \leftarrow OCW(v) - \sum_{u \in C_u} w(v,u)$
end for
return $D$
```

Algorithm 3: Contraction of clusters outside the set of clusters S.

```
Input: $G(V,E)$ and set of clusters $S$
Output: Contracted graph $G'(V',E')$
Add all vertices of $S$ to a new graph $G'$
$\forall i,j \in V' : A'(i,j) \leftarrow A(i,j)$
Add a new vertex $x$ to $G'$
for $\forall i \in \{V' - x\}$ do
    $A'(i,x) = ICW(i) + OCW(i) - \sum_{j \in \{V'-x\}} A'(i,j)$
end for
Obtain $E'$ from $A'$
return $G'(V',E')$
```

If the addition of the new edge does not deteriorate the clustering quality (CASE 1), the clusters are maintained and only the adjacency matrix A and the OCW of the nodes involved are updated. On the other hand if the hypothetical cluster quality of the cluster created by the combination of the two current clusters exceeds a predefined threshold α (CASE 2), the clusters are merged. Otherwise (CASE 3), a new coarsened graph is created by contracting all the clusters except C_u and C_v to a node x. The resulting graph is significantly smaller than the original graph, resulting in lower execution times. An artificial sink t is added to all the vertices of the coarsened graph with weight $\alpha > 0$. After adding edges between t and the other nodes, the minimum-cut tree is calculated, as described in Algorithm 1. The connected components are computed from the resulting tree, after removing t. The components containing vertices of C_u and C_v along with the clusters $C - \{C_v, C_u\}$ are returned as the new clusters of the original graph. As a result of the re-clustering, the ICW and OCW of the nodes involved are recalculated.

4.2.2 Domain-dependent Rules

Additionally domain-dependent rules propagate similarities when clustering occurs. They drive the entire flow of the framework by converting new information into similarities between instances. The four rules that are examined are:

Community. Authors often work together with the same co-author(s). Author instances, that are linked, due to co-authoring, to instances belonging to the same author or authors with similar names, are more similar themselves because of these links. One can distinguish the following three situations ordered by increasing similarity:

- The co-authors of the two instances have similar names.
- The co-authors of the two instances have equal names. An example of this is visualized in Figure 3.
- The co-authors of the two instances are in the same cluster and thus bound to a unique author.

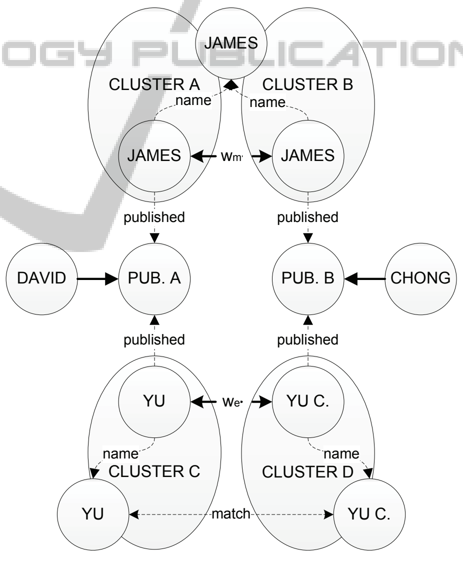

Figure 3: The co-author rule in action: comparing the two instance of James, a similarity (w_m) is added as the co-authors Yu and Yu C. match.

Interest. The topics of publications of the same author are usually located within the same field of research. Keywords extracted from the titles of the papers are defined as the author's interests. There are two dependencies that are considered: time and interest specificity.

The time-dependency increases the precision of this rule by deteriorating the weight of the similarities produced by it as the publication dates are further apart.

The specificity of the interest increases the precision by introducing a dynamic weight as a function of

Table 1: Comparison between the classification of the manually disambiguated family names dataset and the results from DBLP.

Family Name	**Number of Authors**		**Total Number of**
	Manual	**DBLP**	**DBLP Publications**
Turck	4	4	172
Chen	70	1	221
Woo	1	3	9
Mens	2	2	153
Johnson	107	64	460

the number of nouns in a noun phrase. Normally, the longer a noun phrase, the more specific the domain it describes. More specific means that the domain is more characteristic for an author, increasing the probability of two instances working in that domain to be the same author.

e-Mail. Authors with the same e-mail address are most likely to be the same person. The time-dependency is not used in combination with this rule, as e-mails are normally not interchanged between people.

Affiliation. Authors are more likely to work at a specific affiliation at a given time. This rule benefits from the time-dependency concept. Authors sometimes change from one affiliation to another. Having the same affiliation ten years later yields less of a similarity that having the same one right now.

Based on these rules the similarity edge between two instances is assigned a specific weight. This weight is calculated by the disambiguation step that defines priority weights and thresholds ($\alpha > 0$) to compute the probability that the parameters (community, interest, e-mail, affiliation) of the author instances match. Various combinations of priority weights for the four rules are examined in Section 5.1.

Rules are triggered by different events in the system. A rule is for example executed when it has been discovered that an author has published a new publication, but is also executed on the event of re-clustering. The latter is a by-product of the system itself and not originating from an external source. Rules are performed on three different scopes: instances with the same name, instances with similar names and instances that are part of the same cluster. Strictly respecting these scopes narrows down the problem domain.

The clustering process is implemented as a statefull pipe and is completely decoupled from the graph representation. The result is that the graph is almost not being accessed during this process. The grouping of instances is done completely local and the state of the similarities is maintained in a shared key-value store. This approach takes a lot of the load off the repository, which is important as a graph repository does not scale that easily.

5 INITIAL FRAMEWORK EVALUATION

The clustering solution is evaluated on five family names - Turck, Chen, Woo, Mens and Johnson - each with a number of variations. These are manually disambiguated combining the authors into clusters using the information from DBLP and their publications. In total the constructed ground truth test set contains just over 1000 publications. The comparison between the number of different authors obtained after manual disambiguation and these identified by DBLP is presented in Table 1.

The family names of the authors are selected as initial seeds while querying for publications from DBLP. This is combined with e-mail and affiliation information that has been composed manually. The result is a graph containing clusters with the unique author profiles. Next, precision, recall and F-measure (Equation 1) are calculated by comparing the clusters extracted from the graph computed by the minimum-cut tree algorithm using the rules defined in Section 4.2 with the manually composed data set.

$$\begin{aligned} precision &= \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|} \\ recall &= \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|} \\ F_\beta &= (1+\beta^2) * \frac{precision * recall}{\beta^2 * precision + recall} \qquad (1) \end{aligned}$$

5.1 Priority Weights of the Four Rules

Different values for the weights allocated to each of the four rules together with the value of α, which is a threshold used to determine if re-clustering should occur, are tried out in order to verify which combination leads to the most accurate author clustering. Table 2 presents the different distributions of rule

Table 2: Weight distributions for the different rules and the obtained average F-measure over the five family names.

Property	**basic**	**lowkey**	**highco**	**highkey**
α	25	25	25	25
keyword rule	4	1	1	10
community rule	8	8	50	50
affiliation rule	10	10	10	10
email rule	1000	1000	1000	1000
average F-measure	81.5%	72.7%	79.0%	84.5%

weights and the results from clustering are visualized in Figure 4. Four different weight combinations are identified; ”basic” depicts weights resulting in equal priority of all the rules, ”lowkey” decreases the weight associated to the keywords describing the author domain of expertise, ”highco” increases the community rule weight, ”highkey” increases both keyword and community weights.

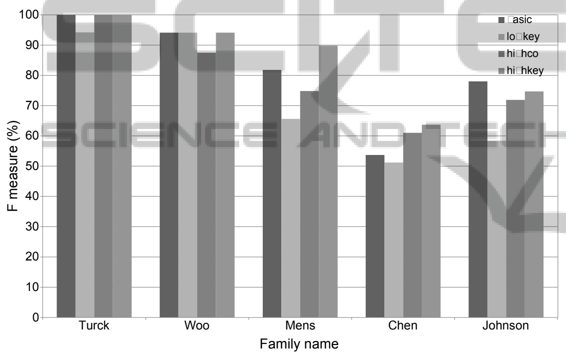

Figure 4: The F-measure for different weight distributions of the rules for each family name.

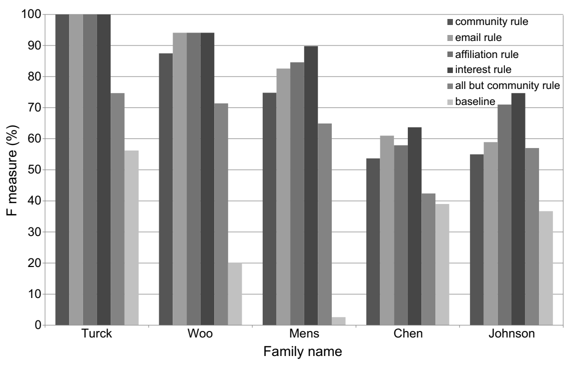

Figure 5: A comparison of different combinations of rules. The first four columns stack the rules, the fifth column uses all rules except the community rule and the last column depicts the baseline, this is the F-measure of the case where no clustering has happened.

The F-measures in Figure 4 show that the ”highkey” distribution provides the best average accuracy. This distribution gives high values to all properties, favouring a lot of clustering, while a high α ensures acceptable threshold. Only for the family name ”Johnson”, it does not get the highest F-measure. This can be attributed to a higher number of clustering iterations, resulting in a higher recall, but a lower precision. The average F-measure for each distribution over the five families is visualized in the last row of Table 2.

5.2 Rule Evaluation

The impact of each of the rules on the accuracy is tested for each of the family names using the ”highkey” weight distribution from the previous section as it yields the highest average accuracy. The F-measure for each of these combinations can be seen in Figure 5. The combination of all four rules generates the best result, although sometimes the increase in accuracy from an additional rule is minimal. In the case of ”Chen”, adding the affiliation rule to the community and e-mail rule even results in a small decrease in accuracy. This is the result of erroneous clustering of certain authors. The community rule on the other hand has the biggest positive impact on the correctness.

The F-measure for each of the family names as partitioned by DBLP is also calculated in order to make a comparison with the presented results in this paper. Table 3 shows that the proposed solution overcomes DBLP by 19% or 28% (weighted based on the number of papers of each author), depending on how the mean accuracy is calculated.

6 CONCLUSIONS

This paper presents an expert-finding repository focusing on author disambiguation by implementing a dynamic minimum-cut tree clustering algorithm. An initial prototype has been developed that continuously gathers author information by identifying initial seeds using a flexible graph as a data model. It incrementally clusters authors based on a domain-independent clustering algorithm and a set of domain-dependent author disambiguation rules assigning similarity weights to author similarities. Validation of the proposed approach shows an improvement on the

Table 3: Comparison of the F-measures for the different family names: DBLP partition vs. the proposed expert repository. The figure also presents the mean F-measure and a weighted (based on the number of papers of each author) F-measure.

%	**Turck**	**Woo**	**Mens**	**Chen**	**Johnson**	**Mean**	**Weighted**
DBLP	100.0	87.5	100.0	2.7	62.8	70.6	61.8
Proposed	100.0	94.1	89.8	63.7	74.7	84.5	79.0
Accuracy Gain	0.0%	+7.5%	-10.2%	+2259.3%	+18.9%	+19.3%	+28.2%

identification of unique author profiles of 28% compared to the results from DBLP.

Future work should focus on enabling the usage of negative weights to the graph model identifying completely different authors. The expansion of the number of online sources in order to retrieve more author information will result in the possible entailment of additional (re)clustering rules.

REFERENCES

Balog, K., Azzopardi, L., and de Rijke, M. (2009). A language modeling framework for expert finding. *Information Processing & Management*, 45(1):1–19.

Balog, K. and de Rijke, M. (2007). Finding similar experts. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 821–822. ACM.

Böhm, C., Naumann, F., et al. (2010). Profiling linked open data with ProLOD. In *Proceedings of the 26th IEEE International Conference on Data Engineering ICDE 2011, Workshops*, pages 175–178.

Boley, H. and Paschke, A. (2007). Expert querying and redirection with rule responder. In *2nd International ExpertFinder Workshop at the 6th International Semantic Web Conference ISWC 2007*.

Fang, H. and Zhai, C. (2007). Probabilistic models for expert finding. *Advances in Information Retrieval*, pages 418–430.

Flake, G., Tarjan, R., and Tsioutsiouliklis, K. (2004). Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408.

Hofmann, K., Balog, K., Bogers, T., and de Rijke, M. (2010). Contextual factors for finding similar experts. *Journal of the American Society for Information Science and Technology*, 61(5):994–1014.

Jung, H., Lee, M., Kang, I., Lee, S., and Sung, W. (2007a). Finding topic-centric identified experts based on full text analysis. In *2nd International ExpertFinder Workshop at the 6th International Semantic Web Conference ISWC 2007*.

Jung, H., Lee, M., Sung, W., and Park, D. (2007b). Semantic Web-Based Services for Supporting Voluntary Collaboration among Researchers Using an Information Dissemination Platform. *Data Science Journal*, 6(0):241–249.

Pavlov, M. and Ichise, R. (2007). Finding experts by link prediction in co-authorship networks. In *2nd International ExpertFinder Workshop at the 6th International Semantic Web Conference ISWC 2007*, pages 42–55.

Pu, K., Hassanzadeh, O., Drake, R., and Miller, R. (2010). Online annotation of text streams with structured entities. In *Proceedings of the 19th ACM international conference on Information and knowledge management CIKM 2010*, pages 29–38.

Saha, B. and Mitra, P. (2006). Dynamic algorithm for graph clustering using minimum cut tree. In *Proceedings of the 6th IEEE International Conference on Data Mining ICDMW '06*, pages 667–671. IEEE.

Sriharee, N. and Punnarut, R. (2007). Constructing Semantic Campus for Academic Collaboration. In *2nd International ExpertFinder Workshop at the 6th International Semantic Web Conference ISWC 2007*, pages 23–32.

Stankovic, M., Jovanovic, J., and Laublet, P. (2011). Linked Data Metrics for Flexible Expert Search on the Open Web. In *Proceedings of the 8th Extended Semantic Web Conference ESWC 2011*, pages 108–123.

Tung, Y., Tseng, S., Weng, J., Lee, T., Liao, A., and Tsai, W. (2010). A rule-based CBR approach for expert finding and problem diagnosis. *Expert Systems with Applications*, 37(3):2427–2438.

Whitelaw, C., Kehlenbeck, A., Petrovic, N., and Ungar, L. (2008). Web-scale named entity recognition. In *Proceeding of the 17th ACM Conference on information and Knowledge Management*, pages 123–132. ACM.

Zhang, J., Tang, J., and Li, J. (2010). Expert finding in a social network. *Advances in Databases: Concepts, Systems and Applications*, pages 1066–1069.

Zhang, J., Tang, J., Liu, L., and Li, J. (2008). A mixture model for expert finding. In *Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, pages 466–478. Springer-Verlag.