Generation of Training Data for Learning-to-Rank Processes in an Expert Seeking Application

Felix Beierle, Felix Engel, Matthias Hemmje

University of Hagen
Multimedia and Internet Applications

felix@beierle.de {felix.engel, matthias.hemmje}@fernuni-hagen.de

Type of work: Master's thesis
Advisors: Dipl.-Inf. Felix Engel, Prof. Dr.-Ing. Matthias Hemmje

Abstract: One of the most important assets in large companies are their experts. Studies in the field of *expertise seeking* have shown that while the specific topic of knowledge is the most relevant aspect in actual searches for experts, other factors regarding the context of the search have to be considered. Within a framework developed at the University of Hagen, *Learning to Rank* is used to learn a ranking function that ranks expertise seeking search results by relevance. This thesis addresses the semi-automatic generation of training data needed for the learning process, employing rules to express relevance patterns.

1 Introduction

Especially in large organizations, the task to find an expert for a specific field is not trivial. Existing applications for the search for experts often only consider topic-factors related to the actual field of expertise. Studies in the field of *expertise seeking* show that while the so-called *quality-related* factors are the most significant in terms of relevance when searching for an expert, there are other important factors, especially contextual factors regarding the relationship of searcher and potential expert, e.g., their familiarity [HBBR10]. In this master's thesis an existing framework developed at the University of Hagen within the SMART VORTEX project (http://www.smartvortex.eu) is used and enhanced. It can retrieve different kinds of information from a semantically annotated knowledge base. The information about the potential experts are stored in a *feature vector* configured by a domain expert. To learn a ranking function to rank the generated feature vectors by relevance, training data has to be provided for the used *Learning to Rank* (LTR) library (different LTR libraries can be used, e.g., RankLib, http://sourceforge.net/p/lemur/wiki/RankLib). Finding reliable training data is an expensive task; so far crowd sourcing, log analysis, or manual specification have been suggested [DFTM12]. This master's thesis follows the idea of [EJH13] to use a system of rules to semi-automatically generate training data for LTR processes. The specific contributions reported here are the analysis of expert seeking parameters that can be considered in such a software (Section 2), and the elaboration and refinement of a system of rules for the comparison of feature vectors on multiple levels (Section 3). As a further part of the thesis, the existing framework is extended to support the retrieval of the needed feature values, and the rule system is implemented.

2 Retrieving Information from the Knowledge Base

Consider the following example: A user of the expert seeking application is assigned to a new project and needs an expert. He is searching for a programming expert to give a presentation on advanced programming techniques with Java and Perl. In this example, the searcher can give the skills 'Java,' 'Perl,' and 'Presentation' as the query.

When searching for an expert, different *expert seeking parameter*s have to be taken into account. *Quality*-related relevance factors are about the formal qualifications of the potential experts [WvdHS12]. ***Topic*** refers to direct links between a user and the asked skills. We use the term ***approach*** bundling aspects regarding the expert's perspective on the asked field of expertise. ***Up-To-Dateness*** refers to temporal information like the last time an asked skill was used for a project. ***Experience*** can include factors like the amount of time someone has been working for the company, years of work experience, number of projects, or the number of connections someone has in the semantically annotated company knowledge base, etc. Studies in expertise seeking come to the conclusion that the familiarity between searcher and expert is the most important relevance factor (with about 10-20%) in the *accessibility* category [WvdHS12, HBBR10]. We refer to space- and time-constraints with the parameter ***proximity*** and to other relational aspects with ***closeness***. In contrast to the quality-related factors, all of the accessibility-related factors depend not only on the expert, but also on the user that is performing the search.

The general approach is as follows: For every person in the knowledge base, a *feature vector* (or *feature value vector*) is constructed, consisting of several *feature*s (e.g. the number of finished related projects). Each feature is represented through a *feature value*. Each expert seeking parameter consists of a set of *relevance aspect*s. One relevance aspect can consist of a single feature (e.g. age). As motivated in [EJH13], there can be dependencies between features, and therefore, one relevance aspect can also consist of more than one feature (e.g. number of projects *and* years of work experience). The framework uses a pairwise LTR approach: Using a system of rules expressing relevance patterns, labeled training data consisting of pairs of feature vectors, along with the information which of the two is to be considered more relevant, is generated. Using the training data, LTR is applied to learn a ranking function by estimating the weight of every component of the feature vectors. The learned ranking model can be used to classify feature vectors from future searches.

Before the software can be used, a *domain expert* has to configure the system. Similar to the idea of an *application context* in [ADM06] we propose that a domain expert with knowledge about the ontology configures the search. In the *feature vector configuration*, he defines the features and how the feature values are calculated. He also defines rules for all relevance aspects, for example: If a person A has completed more projects related to queried skills, and has more years of work experience than person B, then A is more relevant with respect to the relevance aspect 'work experience'. Once the configuration, e.g. for *skill*, of the vector and the rule system is completed, a user may enter a query for a particular set of skills and gets a list of the company's employees sorted by relevance with respect to the queried skills; this list is sorted by the previously learned ranking function.

3 Rule System for Relevance Labeling in a Pairwise LTR Approach

The comparison of two feature vectors $\vec{a}, \vec{b} \in \mathbb{R}^n_{\geq 0}$ can be done with respect to single-feature and multi-feature relevance aspects, and with respect to one or more expert seeking parameters, each consisting of a set of relevance aspects.

Single-Feature Relevance Aspect Comparison Most relevance aspects consist of one single feature (e.g. age). For comparing $\vec{a}$ and $\vec{b}$ with respect to such a single-feature relevance aspect, the corresponding feature values $a_i, b_i \in \mathbb{R}_{\geq 0}$ are compared (the index i indicating the position in the feature vector). Besides providing the two basic comparison operators, greater ($>$) and lesser ($<$), another requirement could be to consider only values that are higher than a certain *threshold* $t \in \mathbb{R}_{\geq 0}$. For instance, looking for an expert with the highest experience, the comparison operator is $>$. A threshold can be used to disregard employees that have not finished at least a certain number of projects. If a company wants to support their younger employees, the relevance aspect 'age' could be used with the comparison operator $<$, using a threshold to disregard employees that are too young.

The comparison yields either T (if a_i is considered more relevant than b_i), F (if b_i is more relevant than a_i), or 0 (neither of a_i, b_i is more relevant than the other one). Thus, the comparison function $c_\diamond(a_i, b_i, t)$ for single-feature relevance aspects with $\diamond$ being $>$ or $<$ and with threshold t has the target set $\{T, F, 0\}$ and is defined as follows:

$$c_>(a_i, b_i, t) = \begin{cases} T & a_i \geq t \wedge a_i > b_i \\ F & b_i \geq t \wedge a_i < b_i \\ 0 & \text{otherwise} \end{cases} \qquad c_<(a_i, b_i, t) = \begin{cases} T & a_i \geq t \wedge a_i < b_i \\ F & b_i \geq t \wedge a_i > b_i \\ 0 & \text{otherwise} \end{cases}$$

Note that both single feature value comparison functions are complementary in the first two arguments, i.e., $c_\diamond(a_i, b_i, t) = T \Leftrightarrow c_\diamond(b_i, a_i, t) = F$, and $c_\diamond(a_i, b_i, t) = 0 \Leftrightarrow c_\diamond(b_i, a_i, t) = 0$, for $\diamond \in \{<, >\}$ and for all non-negative values a_i, b_i, t.

Multi-Feature Relevance Aspects Comparison Following the example of 'work experience,' a person has to have both more years of work experience and a higher number of finished projects to be considered more relevant. In order to compare such multi-feature relevance aspects, several single feature value comparisons have to be taken into account. For this we introduce a three-value logic for the conjunction of two values in $\{T, F, 0\}$: $T \wedge T = T$ and $F \wedge F = F$ and all other conjunctions are evaluated to 0. The idea of this conjunction is that one feature vector has to be more relevant for all single comparisons of the multi-feature comparison to be more relevant with respect to the given multi-feature relevance aspect, for instance:

$$c_>(a_i, b_i, t_1) \land c_>(a_j, b_j, t_2) = \begin{cases} T & c_>(a_i, b_i, t_1) = T \land c_>(a_j, b_j, t_2) = T \\ F & c_>(a_i, b_i, t_1) = F \land c_>(a_j, b_j, t_2) = F \\ 0 & \text{otherwise} \end{cases}$$

Comparison with Respect to Sets of Relevance Aspects For an expert seeking parameter E, let x be the number of comparisons of relevance aspects in E that determine $\vec{a}$ more relevant and let y be the number of comparisons that determine $\vec{b}$ more relevant. If $x > y$, $\vec{a}$ is considered more relevant, if $y > x$, $\vec{b}$ is considered more relevant, otherwise they are considered equally relevant regarding that expert seeking parameter.

Feature Vector Comparison For the comparison of two feature vectors, there is one further level: the aggregation of the results of the comparison with respect to a set of expert seeking parameters. A value between 0 and 1 is assigned to each expert seeking parameter, signifying the percentage of relevance the parameter should take up. The percentages considering a feature vector more relevant are accumulated, and the person with the feature vector rated at a higher cumulated percentage value is labeled as the more relevant person for the given search.

4 Conclusion and Further Work

Within the framework used in this work, we addressed the semi-automatic generation of training data for LTR processes in an expert seeking application, thus avoiding the expensive and time-consuming process of manually generating such data. Future work includes an evaluation of the approach, especially regarding the dependencies between the amount of training pairs, the quality of the ranking model, etc. A further aspect that should be addressed is the implementation of online-learning, where the ranking function is updated through user feedback from previous searches.

References

[ADM06] Riccardo Albertoni and Monica De Martino. Semantic Similarity of Ontology Instances Tailored on the Application Context. In *OTM Conferences*, LNCS Vol. 4275, pages 1020–1038. Springer, 2006.

[DFTM12] Lorand Dali, Blaž Fortuna, Thanh Tran, and Dunja Mladenić. Query-independent Learning to Rank for RDF Entity Search. *The Semantic Web*, pages 484–498, 2012.

[EJH13] Felix Engel, Matthias Juchmes, and Matthias Hemmje. Expert search in semantic annotated enterprise data: integrating query- dependent and independent relevance factors. In *LWA 2013 - Lernen, Wissen & Adaptivität. Workshop Proceedings.*, pages 41–44, Bamberg, 2013.

[HBBR10] Katja Hofmann, Krisztian Balog, Toine Bogers, and Maarten de Rijke. Contextual factors for finding similar experts. *Journal of the American Society for Information Science & Technology*, 61(5):994–1014, 2010.

[WvdHS12] Lilian Woudstra, Bart van den Hooff, and Alexander P. Schouten. Dimensions of quality and accessibility: Selection of human information sources from a social capital perspective. *Information Processing & Management*, 48(4):618–630, 2012.

Additional Information

Bibliographic Data	F. Beierle, F. Engel, and M. Hemmje, “Generation of Training Data for Learning-to-Rank Processes in an Expert Seeking Application,” in *Informatiktage 2014 – Fachwissenschaftlicher Informatik-Kongress, GI-Editionen*. Köllen Durck+Verlag, 2014, pp. 97-100.
Pre-print from	https://beierle.de
Authors	Felix Beierle Felix Engel Matthias Hemmje

BibTeX

```
@inproceedings{Beierle2014Informatiktage,
title = {{Generation of Training Data for Learning-to-Rank Processes in an Expert
Seeking Application}},
author = {Beierle, Felix and Engel, Felix and Hemmje, Matthias},
booktitle = {{Informatiktage 2014 – Fachwissenschaftlicher Informatik-Kongress, GI-
Editionen}},
year = {2014},
volume = {{S-13}},
series = {{LNI}},
pages = {97-100},
publisher = {{K\"ollen Druck+Verlag 2014}},
isbn = {978-3-88579-447-9}
}
```