Finding the Right Experts and Learning to Rank Them by Relevance: Evaluation of a Semi-automatically Generated Ranking Function

Felix Beierle[1,2], Felix Engel[2], Matthias Hemmje[2]

[1] Service-centric Networking, TU Berlin; Telekom Innovation Laboratories
`beierle@tu-berlin.de`
[2] Multimedia and Internet Applications, University of Hagen
{`felix.engel,matthias.hemmje`}`@fernuni-hagen.de`

Abstract. A framework for expert searching developed at the University of Hagen supports the use of contextual factors motivated in the field of expertise seeking. A user can query the system with skills which the person to be found should be an expert in. The result is a list of users from the searched knowledge base, ranked by relevance for the given query. In this paper, we address the semi-automatic generation of training data for the learning-to-rank library that does the relevance ranking. We focus on evaluating the quality of the ranking function.

1 Introduction

In many business situations, it is essential to have the right experts at hand: For instance, for an equipment rental company, failure of equipment is expensive because either downtime has to be financially compensated for or replacements have to be provided to the costumer. In order to do maintenance work on a machine, an engineer needs expertise for a certain machine or device.

Finding such an expert is not a trivial task. Besides his knowledge about the machine/device, the field of *expertise seeking* suggests that contextual factors should be considered, most importantly the familiarity between searcher and expert (e.g. [7]). Through *feature vectors*, potential experts can be represented through numerical values. *Learning to rank* (LTR) can be used to rank those feature vectors by relevance. To learn a ranking function, training data has to be provided. As the generation of training data is expensive, in [6], it is suggested to use a rule-based approach to semi-automatically generate such data. In [4], we briefly reported about the development and implementation of a hierarchical system of rules for the generation of training data. After presenting an overview of this system, the focus of this paper is to evaluate the implemented approach with respect to the quality of the ranking function.

This paper uses results from [3] and is organized as follows: First, we refer to related work (Sec. 2) and give details about the software framework (Sec. 3) that was developed within the SMART VORTEX project (`http://www.smart-vortex.eu`). We sketch the rule system (Sec. 4) introduced in [4] and report about an evaluation (Sec. 5) of the implementation. In Sec. 6 we conclude and point out further work.

2 Related Work

In the following, we will refer to related work in the field of expertise seeking and we will present our concept of ***expert seeking parameters***. When searching for an expert, different expert seeking parameters have to be taken into account. Information about the expert is needed to assess his/her relevance; while this information might be available in a company knowledge base, it may be unknown to the searcher [10]. *Quality*-related relevance factors are about the formal qualifications of the potential experts [11]. ***Topic*** refers to direct links between a user and the asked skills (a skill being, e.g., the knowledge about a specific machine). We use the term ***approach*** bundling aspects regarding the expert's perspective on the asked field of expertise (e.g., engineer vs. construction worker). ***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 a 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 [11] [7]. 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.

Regarding the generation of training data for learning to rank processes, so far, besides manual specification, crowd sourcing [5] and log analysis have been suggested [8]. A less cost intensive approach is to use a rule-based system, as motivated in [6], which we will elaborate in this paper.

3 OWIM SemSearch Framework

At the University of Hagen, the OWIM SemSearch Framework (Open Workbench for Information Management) was developed within the SMART VORTEX project; its architecture is shown in Fig. 1. The general approach is as follows: For every person that is modeled within a semantically annotated knowledge base a *feature vector* (or *feature value vector*) is constructed, consisting of several *features* (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 aspects*. One relevance aspect can consist of a single feature (e.g. age). As motivated in [6], 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 (different LTR libraries can be used, e.g., RankLib, `http://sourceforge.net/p/lemur/wiki/RankLib`): Using a system of rules that express relevance patterns, labeled training data is generated. This data consists of pairs of feature vectors, along with the information which of the two is to be considered more relevant. 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 [2] 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. Details about the implemented algorithms for retrieving data from the ontology, as well as how they are used in the configuration, are given in [3].

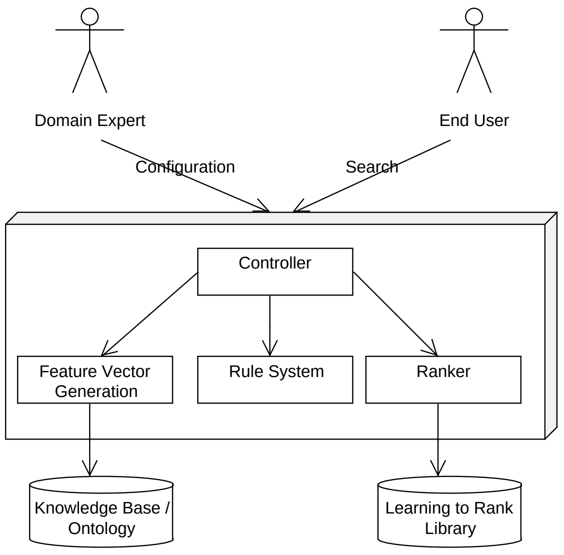

Fig. 1. Architecture

4 Rule System for Relevance Labeling in Pairwise LTR

Single-Feature Relevance Aspect Comparison A relevance aspect can consist of one single feature (e.g. number of publications). For comparing $\boldsymbol{a}$ and $\boldsymbol{b}$ 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 many publications, the comparison operator is $>$. A threshold can be used to disregard employees that have not published at least a certain number of papers. 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.

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 $\boldsymbol{a}$ more relevant and let y be the number of comparisons that determine $\boldsymbol{b}$ more relevant. If $x > y$, $\boldsymbol{a}$ is considered more relevant, if $y > x$, $\boldsymbol{b}$ is considered more relevant, otherwise they are regarded to be equally relevant with respect to 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.

5 Evaluation

The goal of the evaluation is to test the framework with a company knowledge base to determine the factors that play a role regarding the quality of the ranking function, and to check how well the ranking function ranks after learning. Because there is no manually specified ground truth available, the ground truth generated by the rule system will serve as evaluation data. The evaluation is done with the given company knowledge base which has 48 users and 194 skills.

The evaluation is done using k-fold cross validation with Kendall's Tau-b [1]. Kendall's Tau-b is used for the comparison of rankings returned by the rule system and by the learned ranking function, and yields a number between -1 and 1. The value 1 indicates completely concordant rankings, 0 indicates no specific relationship between the two rankings, and -1 indicates completely discordant rankings. The k-fold cross validation yields the *evaluation value*. The higher this value, the better the relevance pattern expressed in the training data could be generalized by the learning to rank library.

First, we will give detailed information about the design of the evaluation and about the factors that have to be considered when evaluating the framework (Section 5.1). Then, we will present and analyze the results of the evaluation in Section 5.2.

5.1 Design

It is expected that the amount of training pairs has the single most significant impact on the evaluation value. In order to show causality and not just correlation, we will take into consideration all factors that could have an impact on the evaluation value. We can distinguish two phases in which variable factors that could impact the evaluation value have to be considered. Firstly, there is the generation of queries with which to generate ground truth. Here, we will introduce four factors ((Q1)-(Q4)) to consider. Secondly, for the evaluation itself, we will introduce three factors to consider ((E1)-(E3)).

Creating Queries For creating queries, the first three factors to consider are:

- **(Q1)** the number of skills in one query
- **(Q2)** the skill(s) in a query
- **(Q3)** the logged-in user who queries the system

To be able to control the amount of created data, we need to limit the amount of users for which feature vectors will be constructed. If there are n users, there are $1/2 * n * (n-1)$ possible pairs of users. When generating ground truth, one user will be logged in and will thus not be in the resulting list of experts. Further considering the number of queries, the following equation (1) shows, how the number of users for which feature vectors are constructed is related to the number of queries and the number of possible training pairs[3]:

$$\begin{pmatrix}\text{number of}\\\text{queries}\end{pmatrix} * \frac{1}{2} * \left(\begin{pmatrix}\text{number of}\\\text{users}\end{pmatrix} - 1\right) * \left(\begin{pmatrix}\text{number of}\\\text{users}\end{pmatrix} - 2\right) = \begin{pmatrix}\text{number of}\\\text{training pairs}\end{pmatrix} \quad (1)$$

Since the used knowledge base contains 48 users, a maximum of $1 * 1/2 * (48 - 1) * (48 - 2) = 1081$ training pairs can be calculated with one query. To get

[3] Both those factors - number of queries and number of training pairs - will be considered later, after the queries are created.

lesser amounts of training data, or to be able to use multiple queries with a fixed amount of pairs to calculate, we will need to limit the *number of users* to create feature vectors for. This gives us an additional item to consider in the creation of queries:

(Q4) the users to create feature vectors for

Macdonald et al. report about *sample selection bias* when choosing samples that are used as training data [9]. Choosing specific data as samples, e.g., data known to be relevant, might influence the learned ranking function to prefer the chosen sample data in subsequent searches. To avoid the sample selection bias, we will choose the factors from the four points (Q1), (Q2), (Q3), and (Q4) completely randomly. The idea of the expert seeking framework is that it should work on any company ontology and regardless of who uses the system. The number of skills per query (Q1) is chosen randomly from 1 to 5. This seems like a reasonable range of skills per query a user will generate.

The queries themselves (the skills used in them) (Q2) and the logged-in user (Q3) are chosen randomly for the evaluation as well. When randomly choosing the logged-in user for the calculation of ground truth, the picked user will be remembered by the framework to pick the same one when comparing the results. In dependence on given amounts of queries and of training pairs, the needed amount of users to calculate feature vectors for is calculated with equation (1) above. After calculating the needed amount of users to calculate feature vectors for, the users are chosen randomly (Q4).

Evaluating For the second phase, the actual evaluation after the creation of ground truth, the first two factors to consider are:

(E1) number of queries
(E2) number of training pairs

To measure a possible impact on the quality of the ranking function, these two factor have to be analyzed. All generated pairs, the ground truth, are split into k groups. Depending on the number of groups, the number of training pairs varies. The number of training pairs is not only dependent on the number of evaluation groups, but also on the number of pairings for which the rule system considers neither feature vector to be more relevant - those pairings cannot be used for training. Both items, the number of queries and the number of training pairs, will be considered in the analysis of the results of the evaluation.

In [9], Macdonald et al. show that too little training data will yield bad results, and that too much training data will take longer to process. Thus, the goal must be to find the lowest number of training pairs that yields good results for the given knowledge base.

Additionally, there is another factor to be considered regarding the configuration of the tests:

(E3) the length of the feature vector

According to the concept presented in Section 2, a minimum of six features will be required to be able to represent every expert seeking parameter. We evaluate three different feature vector configurations, with 6, 9, and 12 features.

The framework is evaluated with two different LTR libraries, RankLib (cf. Section 3) and SVMRank (`http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`). The two LTR libraries only show negligible differences.

5.2 Results

Regarding the number of queries used to generate the ground truth, no correlation can be found between that number and the evaluation value. A higher or lower number of queries does not contribute to a better or worse ranking function.

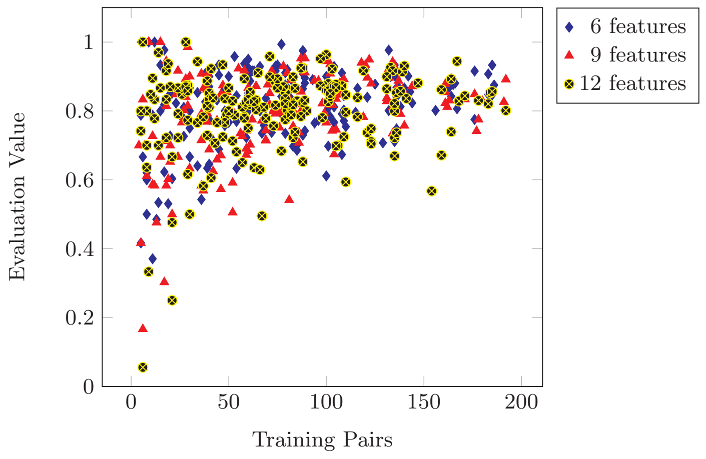

Fig. 2. Number of Training Pairs and Evaluation Value (using SVMRank)

Figure 2 shows the dependencies between the number of training pairs and the evaluation value. Each point in the scatter plot indicates the result of an evaluation. For all given feature vector lengths, roughly the same distribution of points is shown. On the left hand side, for few training pairs, the evaluation value is basically anywhere between 0 and 1. The explanation for this is that with few training pairs, the relevance pattern cannot be properly expressed and overfitting occurs: The learned ranking function is too specific for the training data and does not perform well when classifying unseen data.[4] The more training data is available, the narrower the range of the distribution of points. With enough training data, the relevance pattern properly expressed in the training data can be generalized by the learning to rank library. Generally speaking, for the given knowledge base, at around 150 to 200 training pairs, the evaluation value is always around 0.8.

The length of the feature vector seems to barely have an impact on the evaluation value. For the vector with 12 features, the range of the evaluation value is larger compared to the other feature vector lengths. An explanation for this observation is that the more features are used, the higher the probability for overfitting becomes.

[4] Note that with very few training pairs, occasionally the evaluation value was below 0 while the figure shows only positive results.

6 Conclusions and Future Work

In this paper, we addressed the problem of generating training data for LTR processes in an expert seeking application. The implemented framework can be used to semi-automatically generate training data through a hierarchical system of rules. The evaluation shows that the single most important factor for the quality of the ranking function is the number of training pairs that are used to learn the ranking function. Future work that can be addressed is the implementation of online-learning, where the ranking function is updated through user feedback from previous searches. Our future work also includes using our software with other company knowledge bases, and testing and evaluating the framework with respect to standard information retrieval evaluation methods.

Acknowledgments. This work has been partly supported by the FP7 EU project SMART VORTEX.

References

1. Agresti, A.: Analysis of ordinal categorical data. Wiley series in probability and mathematical statistics: Applied probability and statistics, Wiley (1984)
2. Albertoni, R., De Martino, M.: Semantic similarity of ontology instances tailored on the application context. In: OTM Conferences. pp. 1020–1038. LNCS Vol. 4275, Springer (2006)
3. Beierle, F.: Using a System of Rules to Generate Training Data for Learning-to-Rank Processes in an Expert Seeking Application. Master's thesis, University of Hagen (2014)
4. Beierle, F., Engel, F., Hemmje, M.: Generation of training data for learning-to-rank processes in an expert seeking application. In: Informatiktage 2014 - Fachwissenschaftlicher Informatik-Kongress. Lecture Notes in Informatics (LNI), vol. S-13, pp. 97–100. Köllen Druck+Verlag (2014)
5. Dali, L., Fortuna, B., Tran, T., Mladenić, D.: Query-independent learning to rank for RDF entity search. The Semantic Web pp. 484–498 (2012)
6. Engel, F., Juchmes, M., Hemmje, M.: Expert search in semantic annotated enterprise data: integrating query- dependent and independent relevance factors. In: LWA 2013 - Lernen, Wissen & Adaptivität. Workshop Proceedings. pp. 41–44. Bamberg (2013)
7. Hofmann, K., Balog, K., Bogers, T., Rijke, M.d.: Contextual factors for finding similar experts. Journal of the American Society for Information Science & Technology 61(5), 994–1014 (2010)
8. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 133–142. KDD '02, ACM, New York, NY, USA (2002)
9. Macdonald, C., Santos, R.L.T., Ounis, I.: The whens and hows of learning to rank for web search. Inf. Retr. 16(5), 584–628 (2013)
10. Nevo, D., Benbasat, I., Wand, Y.: The knowledge demands of expertise seekers in two different contexts: Knowledge allocation versus knowledge retrieval. Decis. Support Syst. 53(3), 482–489 (2012)
11. Woudstra, L., van den Hooff, B., Schouten, A.P.: Dimensions of quality and accessibility: Selection of human information sources from a social capital perspective. Information Processing & Management 48(4), 618–630 (2012)