Learning to Rank Experts in Academic Digital Libraries

Catarina Moreira
catarina.p.moreira@ist.utl.pt
Instituto Superior Técnico, INESC-ID
Av. Professor Cavaco Silva, 2744-016 Porto Salvo, Portugal

Abstract

The task of expert finding has been getting increasing attention in the information retrieval literature. However, the current state-of-the-art still lacks in principled approaches for combining different sources of evidence in an optimal way. This article explores the usage of learning to rank methods as a principled approach for combining multiple estimators of expertise, derived from the textual contents, from the graph-structure of the citation patterns for the community of experts, and from profile information about the experts. Several supervised learning to rank algorithms, representative of the pointwise, pairwise and the listwise approaches, were experimented.

This article also explores the effectiveness of rank aggregation approaches combined with data fusion techniques in the task of expert finding in digital libraries. Several experiments were performed with state of the art data fusion techniques.

Experiments made over a dataset of academic publications for the Computer Science domain attest for the adequacy of the proposed approaches.

1 Introduction

The automatic search for knowledgeable people in the scope of specific user communities, with basis on documents describing people's activities, is an information retrieval problem that has been receiving increasing attention [30]. Usually referred to as *expert finding*, the task involves taking a short user query as input, denoting a topic of expertise, and returning a list of people sorted by their level of expertise in what concerns the query topic.

Several effective approaches for finding experts have been proposed, exploring different retrieval models and different sources of evidence for estimating expertise. However, the current state-of-the-art still lacks in principled approaches for optimally combining the multiple sources of evidence that can be used to estimate expertise. In traditional information retrieval tasks such as adhoc retrieval, there has been an increasing interest on the usage of machine learning methods for building retrieval formulas capable of estimating relevance for query-document pairs, as area referred to as Learning to Rank for Information Retrieval (L2R4IR) [22]. The general idea behind traditional L2R4IR approaches is to use hand-labelled data (e.g., document collections containing relevance judgements for specific sets of queries, or information regarding user-clicks aggregated over query logs) to train ranking models, this way leveraging on data to combine the different estimators of relevance in an optimal way. So far, many approaches have been proposed in the expert finding literature, however few previous works have specifically addressed the usage of learning to rank approaches for the task of expert finding. Since hand-labelled data containing relevance judgements are very hard to find for this task, most of the works of the literature are based in probabilistic language models.

This article explores the usage of learning to rank methods, as well as rank aggregation algorithms, in the expert finding task, specifically combining a large pool of estimators for expertise. These include estimators derived from the textual similarity between documents and queries, from the graph-structure of the citation patterns for the community of experts, and from profile information about the experts. We have built a prototype expert finding system using learning to rank algorithms, as well as data fusion techniques, and evaluated it on an academic publication dataset from the Computer Science domain.

The rest of this article is organized as follows: Section 2 presents the main concepts and related works. Section 3 presents the learning to rank approaches used in our experiments. Section 4 details the rank aggregation framework as well as the data fusion techniques used in our experiments. Sec-

tion 5 introduces the multiple features upon which we leverage for estimating expertise. Section 6 describes how the system was evaluated, detailing the datasets used in our experiments as well as the evaluation metrics. Section 7 presents the experimental evaluation of the proposed methods, detailing the obtained results. Finally, Section 8 presents our conclusions and points directions for future work.

2 Concepts and Related Work

Serdyukov and Macdonald have surveyed the most important concepts and representative previous works in the expert finding task [30, 24]. Two of the most popular and well-performing types of methods are the profile-centric and the document-centric approaches [10, 35]. Profile-centric approaches build an expert profile as a pseudo document, by aggregating text segments relevant to the expert [1]. These profiles are latter indexed and used to support the search for experts on a topic. Document-centric approaches are typically based on traditional document retrieval techniques, using the documents directly. In a probabilistic approach to the problem, the first step is to estimate the conditional probability $p(q|d)$ of the query topic q given a document d. Assuming that the terms co-occurring with an expert can be used to describe him, $p(q|d)$ can be used to weight the co-occurrence evidence of experts with q in documents. The conditional probability $p(c|q)$ of an expert candidate c given a query q can then be estimated by aggregating all the evidences in all the documents where c and q co-occur. Experimental results show that document-centric approaches usually outperform profile-centric approaches [1].

Many different authors have proposed sophisticated probabilistic retrieval models, specific to the expert finding task, with basis on the document-centric approach [1, 28, 30]. For instance, Cao et al. proposed a two-stage language model combining document relevance and co-occurrence between experts and query terms[7]. Fang & Zhai derived a generative probabilistic model from the probabilistic ranking principle and extend it with query expansion and non-uniform candidate priors [16]. Zhu et al. proposed a multiple window based approach for integrating multiple levels of associations between experts and query topics in expert finding[45]. Later, Zhu et al. proposed a unified language model integrating many document features for expert finding[44]. Although the above models are capable of employing different types of associations among query terms, documents and experts, they mostly ignore other important sources of evidence, such as the importance of individual documents, or the co-citation patterns between experts available from citation graphs. In this paper, we offer a principled approach for combining a much larger set of expertise estimates for expert finding.

More recently, Macdonald & Ounis proposed a learning to rank approach where they created a feature generator composed of three components, namely a document ranking model, a cutoff value and rank aggregation methods. Using those features, the authors made experiments with the AdaRank listwise algorithm which outperformed all generative probabilistic methods proposed in the literature[25] .

Deng et al. proposed a query sensitive AuthorRank model based on co-authorship networks. In their work, they considered a weighted directed graph to model the co-authorship network, in which each edge represents a co-authorship information. Since the AuthorRank algorithm is query independent, the authors proposed a modification of the this algorithm to take into consideration the query topics when measuring the co-authorship weight between two linked authors[13].

In the Scientometrics community, the evaluation of the scientific output of a scientist has also attracted significant interest due to the importance of obtaining unbiased and fair criteria. Most of the existing methods are based on metrics such as the total number of authored papers or the total number of citations[32, 33]. Simple and elegant indexes, such as the Hirsch index, calculate how broad the research work of a scientist is, accounting for both productivity and impact. Graph centrality metrics inspired on PageRank, calculated over citation or co-authorship graphs, have also been extensively used by Liu et al. [23]. In the context of academic expert search systems, these metrics can easily be used as query-independent estimators of expertise, in much the same way as PageRank is used in the case of Web information retrieval systems.

For combining the multiple sources of expertise, we propose to leverage on previous works concerning the subject of learning to rank for information retrieval (L2R4IR). Liu [22] presented a good survey on the subject, categorizing the previously proposed supervised L2R4IR algorithms into three groups, according to their input representation and optimization objectives:

- **Pointwise approach** - L2R4IR is seen as either a regression or a classification problem. Given feature vectors of each single document from the data for the input space, the relevance

degree of each of those individual documents is predicted with scoring functions which can sort all documents and produce the final ranked list of results. Several different pointwise methods have been proposed in the literature, such as the Additive Groves algorithm proposed by Sorokina et al [36].

- **Pairwise approach** - L2R4IR is seen as a binary classification problem for document pairs, since the relevance degree can be regarded as a binary value which tells which document ordering is better for a given pair of documents. Given feature vectors of pairs of documents from the data for the input space, the relevance degree of each of those documents can be predicted with scoring functions which try to minimize the average number of misclassified document pairs. Several different pairwise methods have been proposed, including SVMrank [21], RankNet [6] and RankBoost [19].

- **Listwise approach** - L2R4IR is addressed in a way that takes into account an entire set of documents, associated with a query, as instances. These methods train a ranking function through the minimization of a listwise loss function defined on the predicted list and the ground truth list. Given feature vectors of a list of documents of the data for the input space, the relevance degree of each of those documents can be predicted with scoring functions which try to directly optimize the value of a particular information retrieval evaluation metric, averaged over all queries in the training data [22]. Several different listwise methods have also been proposed, including SVMmap[42], AdaRank[40, 39] and Coordinate Ascent[26].

In this article, we made experiments with representative supervised learning to rank algorithms from the pointwise, pairwise and listwise approaches. The listwise approaches tested were AdaRank, SVMmap and Coordinate Ascent. The pairwise approaches were RankNet, SVMrank and RankBoost. And Finally, the pointwise approach tested was the Additive Grover algorithm. Section 3 details the considered approaches.

Previous studies have also addressed the problem of L2R4IR through a rank aggregation framework, often with basis on data fusion methods that take their inspiration on voting protocols proposed in the area of statistics and in the social sciences. Riker[29] suggested a classification to distinguish the different existing data fusion algorithms into two categories, namely the positional methods and the majoritarian algorithms.

The positional methods are characterized by the computation of a candidate's score based on the position that the candidate occupies in each ranking lists given by each voter. If the candidate falls in the top of the ranked list, then he receives a maximum score. If the candidate falls in the end of the list, then his score is minimum. The most representative positional algorithms are the borda count fuse[11] and the reciprocal rank fuse[38].

The majoritarian algorithms are characterized by a series of pairwise comparisons between candidates. That is, the winner is the candidate which beats all other candidates by comparing their scores between each other. The most representative majoritarian algorithm is probably the Condorcet fuse method proposed by Montague & Aslam[27]. However, there have been other proposed methods based on Markov chain models[14] as well as techniques from multicriteria decision theory[17].

On the other hand, Fox & Shaw[18] have also proposed another type of data fusion algorithms, based on the aggregation of the scores of documents, which have been widely used in the information retrieval community. Their key contributions comprise the CombSUM algorithm, which will be detailed in Section 4.

3 Learning to Rank Experts

One of the research questions motivating this work was to see if learning to rank approaches can be effectively used in the context of expert search tasks, in order to combine different estimators of expertise in a principled way, this way improving over the current state-of-the art.

In order to validate this hypothesis, a prototype was developed as follows. Given a set of queries $Q = \{q_1, \ldots, q_{|Q|}\}$ and a collection of experts $E = \{e_1, \ldots, e_{|E|}\}$, each associated with specific documents describing the topics of expertise, a training corpus for learning to rank is created as a set of query-expert pairs, each $(q_i, e_j) \in Q \times E$, upon which a relevance judgement indicating the match between q_i and e_j is assigned by a labeller. This relevance judgement is a binary label indicating whether the expert e_j is relevant to the query topics q_i or not. For each instance (q_i, e_j), a feature extractor produces a vector of features that describe the match between q_i and e_j. Features can range from classical IR estimators computed from the documents associated with the experts (e.g., term frequency, inverse document frequency,

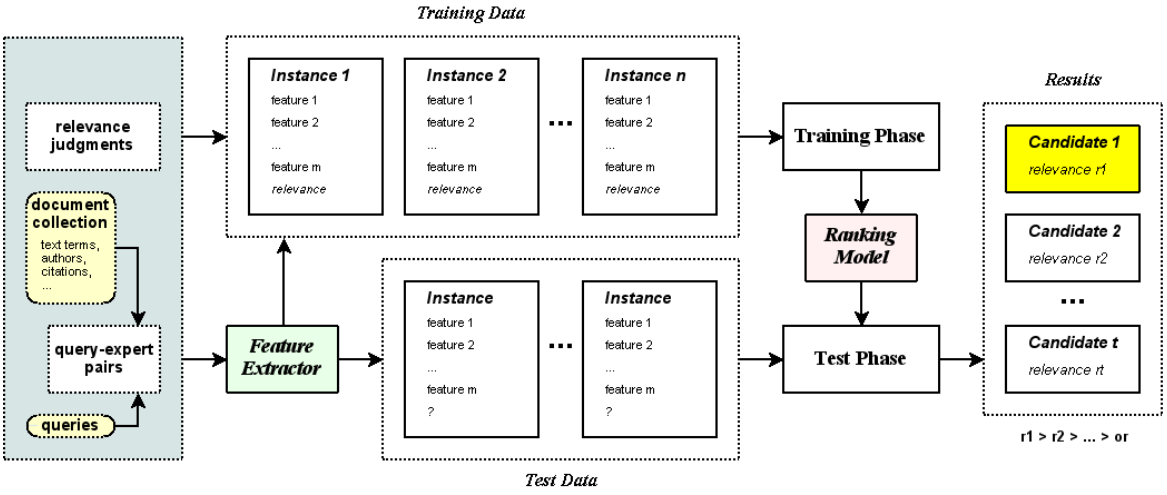

Figure 1: The learning to rank prototype for expert finding in digital libraries

BM25, etc.) to link-based features computed from networks encoding relations between the experts in E (e.g., PageRank). These features are detailed in Section 5 of this article. The inputs to the learning algorithm comprise training instances, their feature vectors, and the corresponding relevance judgements. The output is a ranking function, $f(q_i, e_j)$, which produces a ranking score for each candidate expert e_j so that when sorting experts according to these scores the more relevant ones appear on the top of the ranked list.

During the training process, the learning algorithm attempts to learn a ranking function capable of sorting experts in a way that optimizes a particular bound on an information retrieval performance measure (e.g., Mean Average Precision) for the case of the listwise approaches, or which tries to minimize the number of misclassifications between query-expert pairs, in the case of the pairwise approaches. In the test phase, the learned ranking function is applied to determine the relevance between each expert e_j in E and a new query q. Figure 1 shows a general illustration of the learning to rank prototype built in this work.

In this article, we conducted experiments with representative learning to rank algorithms from the pointwise, pairwise and listwise approaches. The listwise approaches tested were AdaRank, SVMmap and Coordinate Ascent. The pairwise approaches were RankNet, SVMrank and RankBoost. And Finally, the pointwise approach tested was the Additive Grover algorithm. All these algorithms required the manually tuning of parameters in order to give good results. This parameter search was done using a grid search approach in which we place a grid over the parameter space and evaluate the data at every grid intersection, returning the parameters which lead to maximum performance of the learning algorithm[26]. The algorithms used in our experiments are described as follows.

The **AdaRank** listwise method, proposed by Xu et al.[40], builds a ranking model through the formalism of the Boosting approach, attempting to optimize a specific information retrieval performance measure. The basic idea of AdaRank is to train one weak ranker at each round of iteration, and combine these weak rankers as the final ranking function. After each round, the experts are re-weighted by decreasing the weight of correctly ranked experts, with basis on a specific evaluation metric, and increasing the weight of the experts which performed poorly for the same metric. The AdaRank algorithm receives as input the parameter T, which is the number of iterations that the algorithm will perform, and the parameter E, which corresponds to a specific information retrieval performance measure. In the scope of this work, the MAP evaluation metric was used. In our experiments, we set the parameter T to 400.

The **Additive Groves** pointwise method, introduced by Sorokina et al.[36], builds a ranking model through the formalisms of additive models and regression trees, attempting to directly minimize the training errors over the dataset. In this approach, a *grove* is an additive model containing a small number of large trees. The ranking model of a grove is built upon the sum of the ranking models of each one of those trees. The basic idea of Additive Groves is to initialize a grove with a

single small tree. Iteratively, the grove is gradually expanded by adding a new tree or enlarging the existing trees of the model. The trees in the grove are then trained with the set of experts which were misclassified by the other previously trained trees. In addition, trees are discarded and retrained in turn until the overall predictions converge to a stable function. The goal of this algorithm is to find the simplest model which can make the most accurate predictions. The prediction of a grove is given by the sum of the predictions of the trees contained in it. The algorithm receives as input the parameter N, which is the number of trees in the grove, the parameter α, which controls the size of each individual tree and the parameter b which is the number of bagging iterations, i.e. the number of additive models combined in the final ensemble. The parameters were automatically tuned by the algorithm.

The **Coordinate Ascent** listwise method, proposed by Metzler & Croft[26], is an optimization algorithm, used in unconstrained optimization problems, which builds a ranking model by directly maximizing an information retrieval performance measure. The basic idea of Coordinate Ascent is to iteratively optimize a multivariate objective function, by solving a series of one dimensional searches[26]. In each iteration, Coordinate Ascent randomly selects one feature to perform search while holding all other features. This way, in each iteration, the algorithm chooses the parameters which maximize the information retrieval performance measure. In the scope of this work, we used the Mean Average Precision metric as evaluation measure. Equation 1 shows the optimization formula, where E corresponds to the performance measure on the training data (q_i, e_i, y_i).

$$\lambda_i = argmax_{\lambda_i} E(q_i, e_i, y_i) \quad (1)$$

The Coordinate Ascent algorithm receives as input the parameter rr, which is the number of random restarts, and the parameter T, which corresponds to the number of iterations to perform in each one dimensional space. In our experiments, we set the parameter rr to 5 and parameter T to 100.

The **RankBoost** pairwise method, proposed by Freund et al.[19], builds a ranking model through the formalism of the Boosting approach, attempting to minimize the number of misclassified pairs of experts. The basic idea of RankBoost is to train one weak ranker at each round of iteration and combine these weak rankers as the final ranking function. After each round, the expert pairs are re-weighted by decreasing the weight of correctly ranked pairs of experts and increasing the weight of wrongly ranked experts. The RankBoost algorithm receives as input the parameter T, which is the number of iterations that the algorithm will perform, and the parameter θ, which is a threshold corresponding to the number of candidates to be considered in the weak rankers. In our experiments, we set the parameter T to 300 and the parameter θ to 40.

The **RankNet** pairwise method, proposed by Burges et al.[6], builds a ranking model through the formalism of Artificial Neural Networks, attempting to minimize the number of misclassified pairs of experts. The basic idea of RankNet is to use a multilayer neural network with a cost entropy function. While a typical artificial neural network computes this cost by measuring the difference between the network's output values and the respective target values, RankNet computes the cost function by measuring the difference between a pair of network outputs. RankNet attempts to minimize the value of the cost function by adjusting each weight in the network according to the gradient of the cost function. This is done through the usage of the backpropagation algorithm. The RankNet algorithm receives as input the parameter *epochs*, which is the number of iterations through the process of providing the network with an input and updating the network's weights, and the parameter $hiddenNodes$, which corresponds to the number of nodes to be contained in the networks hidden layer. The $hiddenNodes$ parameter has a big impact in the learning process of the network, because if they are too few, we can underfit the data and therefore the network cannot learn any details from the data. On the other hand, if there are too many nodes, we can overfit and therefore the network cannot generalize well and will not be able to predict the value of never seen data. In our experiments, the parameter *epochs* was set to 30 and the parameter $hiddenNodes$ to 50.

The **SVMmap** listwise method, introduced by Yue et al.[42], builds a ranking model through the formalism of structured Support Vector Machines[37], attempting to optimize the metric of Average Precision (AP). The basic idea of SVMmap is to minimize a loss function which measures the difference between the performance of a perfect ranking (i.e., when the Average Precision equals one) and the minimum performance of an incorrect ranking (i.e., when it is less than one). The SVMmap algorithm receives as input the parameter C, which affects the trade-off between model complexity and the proportion of

non-separable samples. If it is too large, we have a high penalty for non-separable points and we may store many support vectors and overfit. If it is too small, we may have underfitting. In our experiments, we used SVMmap with a radial basis function kernel which also requires the manual tuning of the parameter γ, which determines the area of influence of the center support vector has over the data space. In our experiments, we set the parameter C to 8 and the parameter γ to 0.125.

Finally, the **SVMrank** pairwise method, introduced by Joachims[21], builds a ranking model through the formalism of Support Vector Machines. The basic idea of SVMrank is to attempt to minimize the number of misclassified expert pairs. This is achieved by modifying the default support vector machine optimization problem, which considers a set of experts, by constraining the optimization problem to perform the minimization of each pair of experts. This optimization is performed over a set of training queries, their associated pairs of experts and the corresponding relevance judgement over each pair of experts (i.e., pairwise preferences resulting from a conversion from the ordered relevance judgements over the query-expert pairs). SVMrank receives as input the parameter C which affects the trade-off between model complexity and the proportion of non-separable samples. In our experiments, we used a linear kernel, since the radial basis kernel did not converge in convenient time. The parameter C was set to 900.

4 Aggregating Expert Ranks

The general rank aggregation framework for expert finding is illustrated in Figure 2.

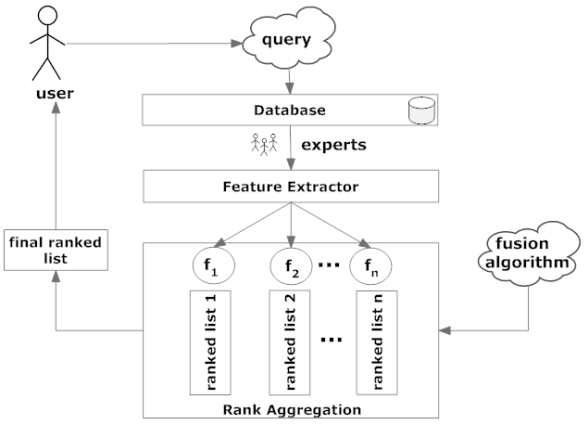

Figure 2: Rank Aggregation Framework for Information Retrieval

Given a set of queries $Q = \{q_1, q_2, ..., q_{|Q|}\}$ and a collection of candidate experts $E = \{e_1, e_2, ..., e_{|E|}\}$ each associated with specific documents describing the candidate's topics of expertise, for each instance (q_i, e_j), a feature extractor produces an ordered ranking list according to each feature that describes the match between q_i and e_j. Features can range from classical IR estimators computed from the documents associated with the experts (e.g., term frequency, inverse document frequency, BM25, etc.) to link-based features computed from networks encoding relations between the experts in E (e.g., PageRank). These features are detailed in Section 5 of this work. A data fusion algorithm is then applied in order to combine the various ranking lists computed by each one of the features. The inputs of a rank aggregation algorithm comprise a set of query-expert pairs corpus, their corresponding feature vectors and the data fusion technique to be applied. The output produces a ranking score resulting from the aggregation of the multiple features. The relevance of each expert e_j towards the query q is determined through this aggregated score.

In this article, we made experiments with representative data fusion algorithms of the information retrieval literature, namely CombSUM, CombMNZ, CombANZ, Borda Fuse, Reciprocal Rank Fuse and Condorcet Fusion algorithms.

All the data fusion techniques applied in the experiments required normalized sums for the different features. So, to perform the normalization, we applied the Min-Max Normalization procedure, which is given by Equation 2.

$$NormalizedValue = \frac{Value - minValue}{maxValue - minValue} \tag{2}$$

The CombSUM, CombMNZ and CombANZ are rank aggregation algorithms originally proposed by Fox & Shaw[18]. These algorithms are defined as follows.

The **CombSUM** score of an expert e for a given query q is the sum of the normalized scores received by the expert in each individual ranking, and is given by Equation 3.

$$CombSUM(e, q) = \sum_{j=1}^{k} score_j(e, q) \tag{3}$$

Similarly, the **CombMNZ** score of an expert e for a given query q is defined by Equation 4, where r_e is the number of systems which contribute to the

retrieval of the candidate.

$$CombMNZ(e,q) = CombSUM(e,q) \times r_e \quad (4)$$

The **CombANZ** score of an expert e for a given query q is defined in the same way as the CombMNZ method, but the scores of the candidates are divided by the number of systems which contribute to the retrieval of the candidate, instead of being multiplied, just like illustrated in Equation 5

$$CombANZ(e,q) = \frac{CombSUM(e,q)}{r_e} \quad (5)$$

The **Borda Fuse** positional rank aggregation method, was originally proposed by de Borda[11] in the scope of voting social theory. The borda fuse determines the highest ranked expert by assigning to each individual candidate a certain number of votes which correspond to its position in a ranked list given by each feature. Generally speaking, if for the feature BM25 candidate e_j appears in the top of the ranking list, it is assigned to him $|E|$ votes, where $|E|$ is the number of experts in the list. If it appears in the second position of the ranked list, it is assigned $|E| - 1$ votes and so on. The final borda score is given by the aggregation of each of the individual scores obtained by the candidate for each individual feature.

The **Reciprocal Rank Fuse** positional rank aggregation method, was originally proposed by Voorches[38] in the scope of the Q&A field. The reciprocal rank fuse determines the highest ranked expert by assigning to each individual candidate a certain score which corresponds to the inverse of its position in a ranked list given by each feature. Generally speaking, if for the feature BM25 candidate e_j appears in the top of the ranking list, it is assigned to him 1/1 votes. If it appears in the second position of the ranked list, it is assigned 1/2 votes and so on. The final reciprocal rank score is given by the aggregation of each of the individual scores obtained by the candidate for each individual feature. The following equation shows the computation of the reciprocal rank scores of some expert e_j using the ranking position of each individual feature, f_i, from a set of features (i=1...n).

$$RR(e_j) = \sum_{i=1}^{n} \frac{1}{position(f_i(e_j))} \quad (6)$$

Finally, the **Condorcet Fusion** majoritarian rank aggregation method, was originally proposed by Montague & Aslam[27] in the scope of the voting social theory. The condorcet fusion method determines the highest ranked expert by taking into account the number of times an expert wins or ties with every other candidate in a pairwise comparison. To rank the candidate experts, we use their win and loss values. If the number of wins of an expert is higher than another, then that expert wins. Otherwise, if they have the same number of wins, then we untie them by their loss scores. The candidate expert with the smaller number of loss scores wins. If the candidates have the same number of wins and losses, then they are tied[4].

5 Features for Estimating Expertise

The considered set of features for estimating the degree of expertise of a researcher towards a given query can be divided into three groups, namely textual features, profile features and graph features. The textual features are similar to those used in standard text retrieval systems and also in previous learning to rank experiments (e.g., TF-IDF and BM25 scores). The profile information features correspond to importance estimates for the authors, derived from their profile information (e.g., number of papers published). Finally, the graph features correspond to importance and relevance estimates computed from the author co-authorship and co-citation graphs.

5.1 Textual Similarity Features

Similarly to previous expert finding works based on document-centric approaches, we also use textual similarities between the query and the contents of the documents to build estimates of expertise. In the domain of academic digital libraries, the associations between documents and experts can easily be obtained from the authorship information associated to the publications. For each topic-expert pair, we used the Okapi BM25 document-scoring function, to compute the textual similarity features. Okapi BM25 is a state-of-the-art IR ranking mechanism composed of several simpler scoring functions with different parameters and components (e.g., term frequency and inverse document frequency). It can be computed through the formula shown in Equation 7, where $Terms(q)$ represents the set of terms from query q, $Freq(i,d)$ is the number of occurrences of term i in document d, $|d|$ is the number of terms in document d, and $\mathcal{A}$ is the average length

of the documents in the collection. The values given to the parameters k_1 and b were 1.2 and 0.75 respectively. Most previous IR experiments use these default values for the k_1 and b parameters.

$$BM25(q,d) = \sum_{i \in Terms(q)} \log\left(\frac{N - Freq(i) + 0.5}{Freq(i) + 0.5}\right) \times \frac{(k_1+1) \times \frac{Freq(i,d)}{|d|}}{\frac{Freq(i,d)}{|d|} + k_1 \times (1 - b + b \times \frac{|d|}{\mathcal{A}})} \quad (7)$$

We also experimented with other textual features commonly used in ad-hoc IR systems, such as *Term Frequency* and *Inverse Document Frequency*.

Term Frequency (TF) corresponds to the number of times that each individual term in the query occurs in all the documents associated with the author. Equation 8 describes the TF formula, where $Terms(q)$ represents the set of terms from query q, $Docs(a)$ is the set of documents having a as author, $Freq(i, d_j)$ is the number of occurrences of term i in document d_j and $|d_j|$ represents the number of terms in document d_j.

$$TF_{q,a} = \sum_{j \in Docs(a)} \sum_{i \in Terms(q)} \frac{Freq(i, d_j)}{|d_j|} \quad (8)$$

The Inverse Document Frequency (IDF) is the sum of the values for the inverse document frequency of each query term and is given by Equation 9. In this formula, $|D|$ is the size of the document collection and $f_{i,D}$ corresponds to the number of documents in the collection where the i_{th} query term occurs.

$$IDF_q = \sum_{i \in Terms(q)} \log \frac{|D|}{f_{i,D}} \quad (9)$$

Other features used were the number of unique authors associated with documents containing the query topics, the range of years since the first and last publications of the author containing the query terms, and the document length, in terms of the number of words, for all the publications associated to the author.

In the computation of these textual features, we considered two different textual streams from the documents, namely (i) a stream consisting of the titles, and (ii) a stream using the abstracts of the articles.

5.2 Profile Information Features

We also considered a set of profile features related to the amount of published materials associated with authors, generally assuming that highly prolific authors are more likely to be considered experts. Most of the features based on profile information are query independent, meaning that they have the same value for different queries. The considered set of profile features are based on the temporal interval between the first and the last publications, the average number of papers and articles per year, and the number of publications in conferences and in journals with and without the query topics in their contents.

5.3 Co-citation and Co-authorship Features

Scientific impact metrics computed over scholarly networks, encoding co-citation and co-authorship information, can offer effective approaches for estimating the importance of the contributions of particular publications, publication venues, or individual authors. Thus, we considered a set of features that estimate expertise with basis on co-citation and co-authorship information. The considered features are divided in two sets, namely (i) citation counts and (ii) academic indexes. In what regards citation counts, we used the total, the average and the maximum number of citations of papers containing the query topics, the average number of citations per year of the papers associated with an author and the total number of unique collaborators which worked with an author. On what regards academic impact indexes, we used the following features:

- **Hirsch index** of the author and of the author's institution, measuring both the scientific productivity and the scientific impact of the author or the institution[20]. A given author or institution has an Hirsch index of h if h of his N_p papers have at least h citations each, and the other $(N_p - h)$ papers have at most h citations each. Authors with a high Hirsch index, or authors associated with institutions with a high Hirsch index, are more likely to be considered experts.

- The ***h*-*b*-index**,which is an extension of the Hirsch index for evaluating scientific topics in general[2]. In the scope of this work, we developed a variation of this index, described as follows. An author has an h-b index of i if i of the N_p papers containing the query terms have at least i citations each, and the other $(Np...i)$ papers have at most i citations each.

- **Contemporary Hirsch index** of the author, which adds an age-related weighting to

each cited article, giving less weight to older articles[31]. A researcher has a contemporary Hirsch index h^c if h^c of his N_p articles get a score of $S^c(i) >= h^c$ each, and the rest $(N_p - h^c)$ articles get a score of $S^c(i) <= h^c$. For an article i, the score $S^c(i)$ is defined as:

$$S^c(i) = \gamma*(Y(now)-Y(i)+1)^{-\delta}*|CitationsTo(i)| \quad (10)$$

In the formula, $Y(i)$ refers to the year of publication for article i. The γ and δ parameters are set to 4 and 1, respectively, meaning that the citations for an article published during the current year account four times, the citations for an article published 4 years ago account only one time, the citations for an article published 6 years ago account 4/6 times, and so on.

- **Trend Hirsch index**[31] for the author, which assigns to each citation an exponentially decaying weight according to the age of the citation, this way estimating the impact of a researcher's work in a particular time instance. A researcher has a trend Hirsch index h^t if h^t of his N_p articles get a score of $S^t(i) >= h^t$ each, and the rest $(N_p - h^t)$ articles get a score of $S^t(i) <= h^t$. For an article i, the score $S^t(i)$ is defined as shown bellow:

$$S^t(i) = \gamma* \sum_{\forall x \in C(i)} (Y(now)-Y(x)+1)^{-\delta} \quad (11)$$

Similarly to the case of the contemporary Hirsch index, the γ and δ parameters are here also set to 4 and 1, respectively.

- **Individual Hirsch index** of the author, computed by dividing the value of the standard Hirsch index by the average number of authors in the articles that contribute to the Hirsch index of the author, in order to reduce the effects of frequent co-authorship with influential authors[3].

- The *a*-**index** of the author or the author's institution, measuring the magnitude of the most influential articles. For an author or an institution with an Hirsch index of h that has a total of $N_{c,tot}$ citations toward his papers, we say that he has an a-index of $a = N_{c,tot}/h^2$.

- The *g*-**index** of the author or his institution, also quantifying scientific productivity with basis on the publication record[15]. Given a set of articles associated with an author or an institution, ranked in decreasing order of the number of citations that they received, the g-index is the (unique) largest number such that the top g articles received on average at least g^2 citations.

- The *e*-**index** of the author[43] which represents the excess amount of citations of an author. The motivation behind this index is that we can complement the h-index by taking into account these excess amounts of citations which are ignored by the h-index. The e-index is given by the Equation 12:

$$e = \sum_{j=1}^{h} \sqrt{cit_j - h^2} \quad (12)$$

In the above equation, cit_j are the citations received by the j_th paper and h is the h-index.

Besides the above features, and following the ideas of Chen et al.[9], we also considered a set of network features that estimate the influence of individual authors using PageRank, a well-known graph linkage analysis algorithm that was introduced by the Google search engine[5].

PageRank assigns a numerical weighting to each element of a linked set of objects (e.g., hyperlinked Web documents or articles in a citation network) with the purpose of measuring its relative importance within the set. The PageRank value of a node is defined recursively and depends on the number and PageRank scores of all other nodes that link to it (i.e., the incoming links). A node that is linked to by many nodes with high PageRank receives a high rank itself.

Formally, given a graph with N nodes $i = 1, 2, \cdots, N$, with L directed links that represent references from an initial node to a target node with weights $\alpha = 1, 2, \cdots, L$, the PageRank Pr_i for the ith node is defined by:

$$Pr_i = \frac{0.5}{N} + 0.5 \sum_{j \in inlinks(L,i)} \frac{\alpha_j Pr_j}{outlinks(L,j)} \quad (13)$$

In the formula, the sum is over the neighbouring nodes j in which a link points to node i. The first term represents the random jump in the graph, giving a uniform injection of probability into all nodes in the graph. The second term describes the propagation of probability corresponding to a random walk, in which a value at node j propagates to node i with probability $\frac{\alpha_j Pr_j}{outlinks(L,j)}$.

The PageRank-based features that we considered correspond to the sum and average of the PageRank values associated to the papers of the author

that contain the query terms, computed over a directed graph representing citations between papers. Each citation link in the graph is given a score of $1/N$, where N represents the number of authors in the paper. Authors with high PageRank scores are more likely to be considered experts.

6 Experimental Validation

The validation of the prototype required a sufficiently large repository of textual contents describing the expertise of individuals. In this work, we used a dataset for evaluating expert search in the Computer Science domain, corresponding to an enriched version of the DBLP[1] database made available through the Arnetminer project.

DBLP data has been used in several previous experiments regarding citation analysis[32, 33] and expert search[12, 13]. It is a large dataset covering both journal and conference publications for the computer science domain, and where substantial effort has been put into the problem of author identity resolution, i.e., references to the same person possibly with different names. Table 1 provides a statistical characterization of the DBLP dataset.

To validate the different experiments performed in this work, it was required a set of queries with the corresponding author relevance judgements. For the Computer Science domain, we used the relevant judgements provided by Arnetminer[2] which have already been used in other expert finding experiments[41, 13]. The Arnetminer dataset comprises a set of 13 query topics from the Computer Science domain, and it was built by collecting people from the program committees of important conferences related to the query topics. Table 2 shows the distribution for the number of experts associated to each topic, as provided by Arnetminer.

To measure the quality of the results produced by the different L2R4IR algorithms and by the different data fusion techniques tested, we used two different performance metrics, namely the Precision at k (P@k) and the Mean Average Precision (MAP).

Precision at rank k is used when a user wishes only to look at the first k retrieved domain experts. The precision is calculated at that rank position through Equation 14.

$$P@k = \frac{r\,(k)}{k} \tag{14}$$

In the formula, $r(k)$ is the number of relevant authors retrieved in the top k positions. $P@k$ only considers the top-ranking experts as relevant and computes the fraction of such experts in the top-k elements of the ranked list.

The Mean of the Average Precision over test queries is defined as the mean over the precision scores for all retrieved relevant experts. For each query r, the Average Precision (AP) is given by:

$$AP[r] = \frac{\sum_{k=1}^{n} P@k[r] \times I\{g_{r_k} = \max(g)\}}{\sum_{k=1}^{n} I\{g_{r_k} = \max(g)\}} \tag{15}$$

In the above equation, n is the number of experts associated with query q and g_{rk} is the relevance grade for author k in relation to the query r. In the case of our datasets, $\max(g) = 1$ (i.e., we have 2 different grades for relevance, 0 or 1).

7 Experimental Results

This section presents the results obtained in both approaches tested in this work, namely the supervised learning to rank approach and the rank aggregation approach for expert finding in digital libraries.

7.1 The Learning to Rank Approach

The main hypothesis behind this work is that learning to rank approaches can be effectively used in the context of expert search tasks, in order to combine different estimators of relevance in a principled way, this way improving over the current state-of-the art. To validate this hypothesis, we have built a prototype expert search system, reusing existing implementations of state-of-the-art learning to rank algorithms. We used the RankLib[3] software package developed by Van Dang, as well as the SVMrank[4] implementation by Joachims[21], the SVMmap[5] implementation by Yue et al.[42] and the Additive Groves algorithm implemented by Sorokina et al.[36]. The statistical tests were performed using ta two sided randomization test made available by Mark D. Smucker[34].

The algorithms contained in the RankLib package are AdaRank, RankNet, RankBoost and Coordinate Ascent. All these algorithms have already been described in Section 3 of this work.

To train and validate the different learning to rank algorithms, since the Arnetminer only contains relevant people for all 13 query topics, it was required to complement this dataset with negative

[1] http://www.arnetminer.org/citation

[2] http://arnetminer.org/lab-datasets/expertfinding/

[3] http://www.cs.umass.edu/~vdang/ranklib.html

[4] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

[5] http://projects.yisongyue.com/svmmap/

Property	Value
Total Authors	1 033 050
Total Publications	1 632 440
Total Publications containing Abstract	653 514
Total Papers Published in Conferences	606 953
Total Papers Published in Journals	436 065
Total Number of Citations Links	2 327 450

Table 1: Statistical characterization of the DBLP dataset used in our experiments

Query Topics	**Rel. Authors**	**Query Topics**	**Rel. Authors**
Boosting (B)	46	Natural Language (NL)	41
Computer Vision (CV)	176	Neural Networks (NN)	103
Cryptography (C)	148	Ontology (O)	47
Data Mining (DM)	318	Planning (P)	23
Information Extraction (IE)	20	Semantic Web (SW)	326
Intelligent Agents (IA)	30	Support Vector Machines (SVM)	85
Machine Learning (ML)	34		

Table 2: Characterization of the Arnetminer dataset of Computer Science experts.

relevant judgements (i.e., adding unimportant authors for each of the query topics). In order to do this, the database was searched with the keywords associated to each topic, retrieving the top $n/2$ authors according to the BM25 metric and retrieving $n/2$ authors randomly selected from the database, where n corresponds to the number of experts associated to each particular topic in the Arnetminer dataset.

In order to make the learning process fair, the feature vectors associated to each candidate were preprocessed, using the package provided by Chakrabarti et al.[8] in the following way:

- **Normalization of the feature vectors**. In the SVM based algorithms and Neural Networks, if the feature vectors are not normalized, then when performing the algorithm, features with high features will tend to ignore the values of very low features, this way leading to an unfair learning process.

- **Removal of null feature vectors**. These vectors do not contribute for the learning process, since they do not provide any additional information to the system.

- **Removal of conflicting feature vectors**. Conflicting feature vectors are the ones which have similar feature values but have opposite relevance judgements. These vectors do not bring additional information to the learning process, since no decision (relevant/non-relevant) can be made with contradictory information.

The test collection was used in a leave-one-out cross-validation methodology, in which different experiments used 9 different queries to train a ranking model, which was then evaluated over the remaining queries. The averaged results from the four different cross-validation experiments were finally used as the evaluation result.

Table 3 presents the obtained results over the DBLP dataset. These results show that the pointwise approach Additive Groves outperformed all other pairwise and listwise learning to rank algorithms, in terms of MAP, and therefore provided a better classification and ranking procedure than all other approaches tested. On the other hand, the listwise SVMmap algorithm, with a RBF kernel, performed almost as good as the Additive Groves method. This makes sense, since the goal of SVMmap is to optimize the Mean Average Precision scores. In fact, SVMmap outperformed Additive Groves when ranking the top 5 experts (P@5), showing that this listwise method can also be successfully used in the context of expert finding.

Table 3 also shows competitive results for the pairwise approaches SVMrank, with a linear kernel, and RankBoost. This leads to the conclusion that pairwise learning to rank algorithms based on the formalisms of support vector machines and on the boosting framework can also be applied successfully in the task of expert finding.

As for the worst results obtained, Table 3 shows

that AdaRank and RankNet were not as successful as the other algorithms. For RankNet, an easy explanation can be found in the usage of the gradient descent in the backpropagation algorithm. Gradient descent is an optimization algorithm which tries to find the minimum of a function by taking steps proportional to the negative gradient of the function. However, this method has the problem of not being able to find a global minimum, because it can get stuck in a local one and therefore a good optimization may never be achieved. In addition, the cost function used in RankNet is general and does not correspond to any specific information retrieval metric. For the case of AdaRank, its greedy feature selection method did not perform very well in this experiment. At this moment, one might be thinking why AdaRank performed poorly, if the RankBoost algorithm performed quite good, since they are both based in the same Boosting formalisms? One explanation can be found in the extra parameter which is present in the RankBoost approach, namely the number of threshold candidates. AdaRank is only focused in making a greedy linear combination of weak rankers, while RankBoost, not only makes a linear combination, but also takes into consideration if an expert returned by a query is above that threshold, this way decreasing the weight of the weak rankers of correctly ranked experts and consequently building more accurate ranking functions.

Finally, Table 3 shows that all the algorithms tested in this supervised framework outperformed the baseline ranking function BM25, showing the adequacy of all the learning to rank algorithms for the expert finding task in digital libraries.

When using learning to rank approaches for information retrieval, it is important to perform statistical tests in order to determine whether an algorithm actually outperforms another. In the scope of the supervised learning to rank experiments performed in this work, we applied a two sided paired randomization test in order to see if the best performing algorithm, more specifically Additive Groves, was indeed more significant than all other algorithms. The test results indicated that the Additive Groves algorithm, in terms of MAP, was more significant than RankNet, RankBoost, Coordinate Ascent and AdaRank for a confidence interval of 95%. However, it was not more significant than the approaches based on the formalisms of Support Vector Machines, more specifically SVMrank and SVMmap, since their significance levels were high (0.0893 for SVMrank and 0.3370 for SVMmap).

In a separate experiment, we attempted to measure the impact of the different types of ranking features on the quality of the results. Using the best performing learning to rank algorithm, namely the Additive Groves method, the results obtained were measured by ranking models which considered (i) only the textual similarity features, (ii) only the profile features, (iii) only the graph features, (iv) textual similarity and profile features, (v) textual similarity and graph features and (vi) profile and graph features. Table 4 shows the obtained results, also presenting the previous results reported by Yang et al.[41], as well as the representative state of the art approaches proposed by Balog et al.[1], over the same dataset.

As one can see, the set with the combination of all features has the best results. The results also show that the combination of profile features with graph features have the poorest results. This means that the presence of the query topics in the author's publications, namely in the titles and abstracts, is crucial to determine if some authors are experts or not, and indeed the information provided by textual evidences can help in expertise retrieval. Finally, the results show that the different combinations of all features proposed in this paper outperform the previously learning to rank approach of the literature proposed by Yang et al[41].

Table 4 also presents two state of the art approaches for the task of expert finding on an organization environment. These approaches are the baseline models proposed by Balog et al., namely the candidate-based Model 1 and the document-based Model 2[1]. These results were reported by Deng et al.[13] which have modified the original code developed by Balog et al. in order to work for the DBLP dataset. This way, it is possible to make a fair comparison between a representative approach of the state of the art and the supervised learning to rank approach proposed in this work. As one can see, Balog's Model 1 achieved very low results. One explanation is that when an author publishes a paper which contains a set of words which exactly match the query topics, this author achieves a very high score in Model 1. In addition, since we are dealing with very big datasets, there are lots of authors in such situation and consequently the top ranked authors are dominated by non-experts, while the real experts will be ranked lowered. Model 2, on the other hand, is more suitable for the task of expert finding in digital libraries, since the candidates are ranked by aggregating the relevant papers of each one of them. Nevertheless, all the supervised learning to rank algorithms proposed in this work outperformed all representative state of the art approaches.

Taking into consideration the best performing approach, namely the Additive Groves algorithm with the full set of features, we applied a two sided

	P@5	P@10	P@15	P@20	MAP	NDCG
AdaRank	0.6667	0.6834	0.6736	0.6813	0.6478	0.8848
Coordinate Ascent	0.9250	0.8729	0.8417	0.8250	0.7577	0.9361
RankNet	0.7042	0.7187	0.6875	0.6761	0.6530	0.8726
RankBoost	0.8375	0.8792	0.8320	0.8146	0.7840	0.9395
Additive Groves	0.9667	**0.9583**	**0.9472**	**0.9396**	**0.8940**	**0.9734**
SVMmap	**0.9875**	0.9208	0.8972	0.8917	0.8702	0.9700
SVMrank	0.9542	0.9125	0.8820	0.8833	0.8311	0.9561
BM25 (baseline)	0.6923	0.5769	0.5026	0.4769	0.5422	0.8382

Table 3: Results the various learning to rank algorithms tested

	P@5	P@10	P@15	P@20	MAP	NDCG
Text Similarity + Profile + Graph	**0.9667**	**0.9583**	**0.9472**	**0.9396**	**0.8940**	**0.9734**
Text Similarity + Profile	**0.9667**	0.9438	0.9362	0.9167	0.8714	0.9634
Text Similarity + Graph	0.9334	0.9438	0.9361	0.9240	0.8825	0.9687
Profile + Graph	0.9208	0.8750	0.8625	0.8500	0.8237	0.9458
Text Similarity	0.9334	0.9188	0.9125	0.9010	0.8660	0.9645
Profile	**0.9667**	0.9104	0.9153	0.9125	0.8728	0.9665
Graph	0.8917	0.9167	0.8931	0.8948	0.8526	**0.9734**
Balrog's Model 1[1]	-	0.0250	-	0.0180	0.0060	-
Balrog's Model 2[1]	-	0.5750	-	0.4600	0.3915	-
Expert Finding [41]	0.5500	0.6000	0.6333	-	0.6356	-

Table 4: The results obtained with different sets of features and comparison with other approaches.

pair randomization test in order to determine if this best approach was indeed more significant than the Additive Groves algorithm using (i) textual similarity features, (ii) graph features, (iii) profile features, (iv) profile and graph features, (v) text and graph features and (vi) text and profile features, in terms of MAP. The results indicated that the Additive Groves combined with the set of all features was more significant than the Additive Groves algorithm using (i) profile and graph features, (ii) text and graph features and (iii) text and profile features, for a confidence interval of 95%. However, the same algorithm with the full set of features was not more statistically significant than Additive Groves using (i) textual features, (ii) profile features and (iii) graph features, since their significance levels were very high (0.0814 for textual features, 0.5359 for profile features and 0.1878 for graph features).

Figure 3 plots the obtained average precision in each of the individual query topics for the best performing approach, namely Additive Groves with the full set of features. The figure presents the query topics in the same order as they are given in Table 2. The horizontal dashed line corresponds to the MAP obtained in the same experiment. The results show that there are only slightly variations in performance for the different queries.

Finally, Table 5 shows the top five people which were returned by the system for four different queries, corresponding to the best and worst results in terms of the P@5 metric. The system performed well for the queries Neural Networks, Computer Vision and Cryptography. However, worse results were returned for the query Information Extraction. Since the textual features have a lot of impact in the classification process, these poor results can be explained by the general use of the query terms. Many works from the various fields of computer science contain the terms *information* and *extraction*. One reason for the misclassifications found for this query are due to the way we constructed the non-relevant authors data. When selecting the non-relevant authors for the information extraction query, many of them could belong from other areas of the computer science domain, not necessarily belonging to the information extraction field. For instance, the top expert returned, Mark Craven, works in bioinformatics has many publications with the word *information*. Another example is the second retrieved author, Reinhard C. Laubenbacher, which works in algorithms and algebra. Again the word *information* appears several times in this publications' titles and abstracts contributing for a misclassification.

7.2 The Rank Aggregation Approach

Since relevant judgements for the expert finding task in digital libraries are very hard to find, we decided to perform rank aggregation experiments

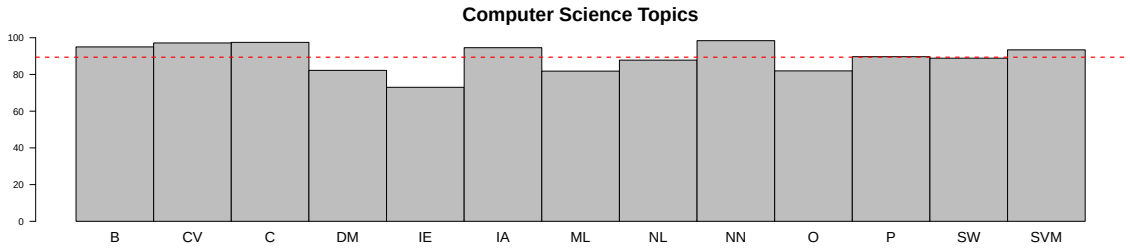

Figure 3: Average precision over the different query topics.

Best Results			Worst Results
Neural Networks	**Computer Vision**	**Cryptography**	**Information Extraction**
Geoffrey E. Hinton	Bernard F. Buxton	Johannes Buchmann	Mark Craven
Erkki Oja	Ioannis A. Kakadiaris	Phillip Rogaway	Reinhard C. Laubenbacher
Yann LeCun	Daphna Weinshall	David Chaum	Jude W. Shavlik
Thomas G. Dietterich	Louise Stark	Susanne Wetzel	Ellen Riloff
Michael I. Jordan	Shimon Ullman	Tal Rabin	Remzi H. Arpaci-Dusseau

Table 5: Top five people returned by the system for four different queries.

using data fusion techniques. Our hypothesis is that, although rank aggregation techniques are far away to perform as accurately as the supervised approaches, rank aggregation approaches can also provide acceptable results in this task, outperforming representative approaches of the literature.

To validate this hypothesis, we have built a prototype expert search system by implementing 6 different data fusion algorithms based in positional methods, majoritarian methods and score aggregation. The score aggregation algorithms implemented were CombSUM, CombMNZ and CombANZ. The positional algorithms were the Borda Fuse and Reciprocal Rank Fuse. And the majoritarian algorithm implemented was the Condorcet fusion. All these algorithms have been described in Section 4 of this work.

To validate the different rank aggregation algorithms, since the Arnetminer only contains relevant people for all 13 query topics, it was required to complement this dataset with negative relevant judgements (i.e., adding unimportant authors for each of the query topics). In order to do this, the database was searched with the keywords associated to each topic, keeping up to 350 authors that were either marked as relevant or that were highly ranked according to the BM25 metric, thus simulating a real search scenario.

Table 6 presents the obtained results over the DBLP dataset. The obtained results show that the CombMNZ rank aggregation technique outperformed all other algorithms, in terms of MAP, showing that this rank aggregation method provides a better ranking than all other approaches. On the other hand, the Condorcet Fusion algorithm outperformed all other methods in almost all the evaluation metrics tested. In fact, the Condorcet Fusion achieved much better results for the P@k than all other algorithms. P@k is an important evaluation metric in expert finding, because when the user searches for experts of some topic, he is only interested in the top 20, at most 50, experts. So, although the Condorcet Fusion algorithm did not outperform CombMNZ in terms of MAP, it is the approach tested with the best overall results.

Table 3 also shows that the Borda Fuse and the Reciprocal Rank Fuse algorithms had the same performance in our experiments. This is not surprising, because these positional algorithms are very similar. The only difference between each other is that Borda Fuse uses directly the position of the candidates, whereas Reciprocal Rank Fuse uses the reciprocal rank of the position of the candidates. It turns out that, in this experiment, the final ranked lists returned were the same.

As for the worst results obtained, Table 3 shows that CombANZ was not as successful as the other algorithms. This can be explained by the fact that the CombANZ algorithm divides the CombSUM scores by the number of systems which contribute for the ranking of a candidate. This is not a very good approach because of the following example. Suppose that candidate e_1 has a CombSUM score of 1 and that 5 systems contributed for his ranking. So his CombANZ score is 0.2. Now, suppose another candidate e_2 with a CombSUM score also 1, but the

number of systems contributing for its ranking are only 2. So his CombANZ score is 0.5. Therefore, candidate e_2 is given more importance than candidate e_1. This is not a good ranking, since candidate e_1 had more systems supporting his score.

Finally, Table 3 shows that all the algorithms tested in this rank aggregation framework outperformed the baseline ranking function BM25, showing the adequacy of all these data fusion algorithms for the expert finding task in digital libraries.

We also applied a two sided paired randomization test in order to see if the best performing algorithm, more specifically Cordocet Fuse, was indeed more significant than all other algorithms. The test results indicated that the Condorcet Fusion algorithm, in terms of MAP, was more significant than the baseline BM25 function and CombANZ for a confidence interval of 95%. However, Cordorcet Fuse was not more significant than the Borda Fuse, Reciprocal Rank Fuse, CombMNZ and CombSUM algorithms, since their significance levels were high (0.2005 for Borda Fuse and Reciprocal Rank Fuse, 0.6213 for CombMNZ and 0.3094 for CombSUM).

In a separate experiment, we attempted to measure the impact of the different types of ranking features on the quality of the results. Using the Condorcet Fusion algorithm, we measured the results obtained by ranking models that considered (i) only the textual similarity features, (ii) only the profile features, (iii) only the graph features, (iv) textual similarity and profile features, (v) textual similarity and graph features and (vi) profile and graph features. Table 7 shows the obtained results, also presenting the previous results reported by Yang et al.[41] and Deng et al.[13], as well as the representative state of the art approaches proposed by Balog et al.[1], over the same dataset.

As one can see, the set with the combination of all features had the best results. Since DBLP has rich information about citation links, one can see that the set of graph features achieved the best results for this dataset in terms of MAP. The results also show that, individually, textual similarity features have the poorest results. This means that considering only textual evidence provided by query topics, together with article's titles and abstracts, may not be enough to determine if some authors are experts or not, and that indeed the information provided by citation and co-authorship patterns can help in expertise retrieval in a rank aggregation framework. Finally, the results show that the different combinations of all features proposed in this paper outperformed the representative state of the art approaches proposed by Balog et al.[1], namely the candidate-based Model 1 and the document-based Model 2. On the other hand, the rank aggregation approach proposed in this work has a slightly poor performance when compared to the query sensitive AuthorRank model proposed by Deng et al[13]. One reason for these results is that Deng et al. first uncovered authors associated with communities, that is, they first determined which authors were publishing, for instance, in Neural Networks conferences or journals. Then, given a query topic, their model tries to determine which are the top communities associated to the query and then they extract the authors. Consequently, their approach has a high probability of returning authors which, in fact, work in the field of the query topics, making their system more reliable and accurate. Table 7 also shows that this rank aggregation framework did not outperform the previously learning to rank approach proposed by Yang et al.[41]. This was already expected, since supervised approaches find optimal and more accurate ranking models than the rank aggregation approaches. However, one must take into consideration that it is very hard to find a sufficiently large training corpus for the expert finding task domain in digital libraries, which difficult the usage of machine learning approaches in this area, and therefore rank aggregation approaches are preferable. Nevertheless, the rank aggregation approach proposed in this work also provides very competitive results for the expert finding literature.

Taking into consideration the best performing approach, namely the Condorcet Fusion algorithm with the set of all features, we applied a two sided pair randomization test in order to determine if this best approach was indeed more significant than the Condorcet Fusion algorithm using (i) textual similarity features, (ii) graph features, (iii) profile features, (iv) profile and graph features, (v) text and graph features and (vi) text and profile features, in terms of MAP. The results indicated that Condorcet Fusion combined with the set of all features was more significant than the Condorcet Fusion algorithm using (i) textual similarity features, (ii) profile, (iii) text and graph features and (iv) text and profile features, for a confidence interval of 95%. However, the Cordorcet Fuse algorithm with the full set of features was not more significant than the same algorithm using (i) graph features and (ii) profile and graph features, since their significance levels were very high (0.6245 for graph features and 0.0779 for the graph and profile features).

	P@5	P@10	P@15	P@20	MAP	NDCG
CombSUM	0.4000	0.4077	0.4154	0.4500	0.4134	0.7385
CombMNZ	0.4923	0.4769	0.4718	0.5115	**0.4843**	0.7757
CombANZ	0.3077	0.3231	0.3795	0.4000	0.3561	0.6985
Borda Fuse	0.2000	0.2231	0.3026	0.3423	0.3999	0.7011
Reciprocal Rank Fuse	0.2000	0.2231	0.3026	0.3423	0.3999	0.7011
Condorcet Fusion	**0.7077**	**0.6077**	**0.5641**	**0.5154**	0.4382	**0.7975**
BM25 (baseline)	0.4923	0.4308	0.3846	0.3385	0.3264	0.7094

Table 6: Results the various learning to rank algorithms tested

	P@5	P@10	P@15	P@20	MAP	NDCG
Text Similarity + Profile + Graph	**0.7077**	0.6077	**0.5641**	0.5154	0.4382	**0.7975**
Text Similarity + Profile	0.4000	0.4154	0.4000	0.3731	0.3267	0.7042
Text Similarity + Graph	0.5692	0.5231	0.4769	0.4500	0.3908	0.7616
Profile + Graph	0.6308	0.5385	0.4821	0.4615	0.4165	0.7744
Text Similarity	0.3500	0.3333	0.3444	0.3125	0.2975	0.6681
Profile	0.4615	0.4308	0.4154	0.4192	0.3687	0.7242
Graph	0.6462	0.5769	0.5436	0.5308	0.4386	0.7857
Balrog's Model 1[1]	-	0.0250	-	0.0180	0.0060	-
Balrog's Model 2[1]	-	0.5750	-	0.4600	0.3915	-
Query-Sensitive AuthorRank[13]	-	**0.6800**	-	0.5350	**0.4906**	-
Expert Finding [41]	0.5500	0.6000	0.6333	-	0.6356	-

Table 7: The results obtained with different sets of features and comparison with other approaches.

8 Conclusions and Future Work

With this work, we showed that learning to rank approaches perform very well in the task of expert finding in digital libraries. In fact, they outperform the approach proposed by Yang et al.[41] which, as far as we know, is the only one which tackles the problem of expert finding as a learning to rank approach for the DBLP dataset.

The various learning algorithms tested showed significant different results between them, which makes us conclude that some algorithms are more suitable for this task than others. In addition, we experimentally demonstrated that the Additive Groves pointwise approach and the SVMmap listwise approach outperformed all other algorithms and consequently are the most suitable learning to rank approaches for the expert finding task in digital libraries. These results were quite interesting since pointwise approaches do not take into consideration the order of the experts and therefore the worst results were expected by this approach. However, one must take into consideration that the Additive Groves algorithm is very robust in a way that, in each iteration, it always trains a new and more accurate model for the experts which were misclassified by the previous one. In addition, this algorithm was amongst the top 5 winners of the Yahoo! Learning to Rank Competition[6]. This leads to the conclusion that pointwise approaches can also be very effective, just like the listwise ones, as long as they are carefully designed.

In what concerns our rank aggregation approach, the results showed that rank aggregation approaches also provide reasonable results for the task of expert finding in digital libraries, since they outperformed some state of the art works, in terms of MAP. These rank aggregation approaches are far away to perform as successfully as the supervised ones, however one must take into consideration that it is very hard to find hand labelled data describing relevant people in some topic, and therefore these approaches have been preferable, in the expert finding literature, than supervised ones. In our experiments, CombMNZ and the Condorcet Fusion algorithm achieved the best results, which leads to the conclusion that majoritarian methods, as well as rank aggregation algorithms, are the most suitable for the task of expert finding in digital libraries.

Since the learning to rank and the rank aggregation approaches performed very well, always outperforming representative works of the state of the art, then it is straightforward the effectiveness of the features proposed in this article. When comparing all different sets of features, we concluded that the combination of all features (textual, profile and graph) were required in order to achieve the best results in both experiments.

[6] `http://learningtorankchallenge.yahoo.com/`

As for future work, it would be very interesting to apply the algorithms tested in this work in the TREC enterprise task dataset. In the learning to rank approach for expert finding proposed by Macdonald & Ounis[25], they achieved best results using the AdaRank listwise algorithm, turning their approach one of the top contributions for the expert finding task in enterprises. However, the experiments performed in the scope of this article showed that AdaRank was the approach which performed poorly. We are very curious to know how the Additive Groves algorithm would perform in such task.

References

[1] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th annual international ACM Conference on Research and Development in Information Retrieval (SIGIR '06)*, 2006.

[2] Michael G. Banks. An extension of the hirsch index: Indexing scientific topics and compounds. *Scientometrics*, 69:161–168, 2006.

[3] Pablo D. Batista, Mônica G. Campiteli, and Osame Kinouchi. Is it possible to compare researchers with different scientific interests? *Scientometrics*, 68:179–189, 2006.

[4] Ilker Nadi Bozkurt, Hayrettin Gurkok, and Eyup Serdar Ayaz. Data fusion and bias, 2007.

[5] Sergey Brin, Lawrence Page, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford Digital Library Technologies Project, 1999.

[6] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, 2005.

[7] Yunbo Cao, Jingjing Liu, Shenghua Bao, and Hang Li. Research on expert search at enterprise track of trec 2005. In *Proceedings of the 14th Text REtrieval Conference (TREC 2005)*, 2006.

[8] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. Structured learning for non-smooth ranking losses. In *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD '08)*, 2008.

[9] Pon-Jen Chen, Huafeng Xie, Sergei Maslov, and Sidney Redner. Finding scientific gems with google's pagerank algorithms. *Journal of Informetrics*, 1(1):8–15, 2007.

[10] Nick Craswell, Arjen P. de Vries, and Ian Soboroff. Overview of the trec-2005 enterprise track. In *Proceedings of the 14th Text REtrieval Conference (TREC 2005)*, 2005.

[11] Jean-Charles de Borda. *Mémoire sur les Élections au Scrutin.* Histoire de l'Académie Royale des Sciences., 1781.

[12] Hongbo Deng, Irwin King, and Michael R. Lyu. Formal models for expert finding on dblp bibliography data. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*, 2008.

[13] Hongbo Deng, Irwin King, and Michael R. Lyu. Enhanced models for expertise retrieval using community-aware strategies. *IEEE Transactions on Systems, Man, and Cybernetics*, 2011.

[14] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation revisited. In *Proceeding of the 10th World Wide Web Conference Series*, 2001.

[15] Leo Egghe. Theory and practise of the g-index. *Scientometrics*, 69(1):131–152, 2006.

[16] Hui Fang and ChengXiang Zhai. Probabilistic models for expert finding. In *Proceedings of the 29th European Conference on Information Retrieval Research (ECIR '07)*, 2007.

[17] Mohamed Farah and Daniel Vanderpooten. An outranking approach for rank aggregation in information retrieval. In *Proceedings of the 30th annual international ACM Conference on Research and Development in Information Retrieval (SIGIR '07)*, 2007.

[18] Edward Fox and Joseph A. Shaw. Combination of multiple searches. In *Proceedings of the 2nd Text Retrieval Conference (TREC 1994)*, 1994.

[19] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[20] J. E. Hirsch. An index to quantify an individual's scientific research output. In *Proceedings of the National Academy of Sciences USA (PNAS '05)*, 2005.

[21] Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD '06)*, 2006.

[22] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations of Trends Information Retrieval*, 3:225–331, 2009.

[23] Xiaoming Liu, Johan Bollen, Michael L. Nelson, and de Herbert Van de Sompel. Co-authorship networks in the digital library research community. In *Information Processing and Management*, 2005.

[24] Craig Macdonald and Iadh Ounis. Voting techniques for expert search. In *Knowledge Information Systems*, 2008.

[25] Craig Macdonald and Iadh Ounis. Learning models for ranking aggregates. In *Proceedings of the 33rd European Conference on Information Retrieval (ECIR '11)*, 2011.

[26] Donald Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 16(28):1–23, 2007.

[27] Mark H. Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *Proceedings of the 11th international conference on information and knowledge management (CIKM '02)*, 2002.

[28] Desislava Petkova and Bruce Croft. Proximity-based document representation for named entity retrieval. In *Proceedings of the 16th ACM conference on Conference on information and knowledge management (CIKM '07)*, 2007.

[29] William H. Riker. *Liberalism Against Populism: A Confrontation Between the Theory of Democracy and the Theory of Social Choice*. Waveland Press, 1988.

[30] Pavel Serdyukov. *Search for Expertise : Going Beyond Direct Evidence*. PhD thesis, University of Twente, 2009.

[31] Antonis Sidiropoulos, Dimitrios Katsaros, and Yannis Manolopoulos. Generalized h-index for disclosing latent facts in citation networks. *Scientometrics*, 72(2):253–280, 2007.

[32] Antonis Sidiropoulos and Yannis Manolopoulos. A citation-based system to assist prize awarding. *SIGMOD Record*, 34:54–60, 2005.

[33] Antonis Sidiropoulos and Yannis Manolopoulos. Generalized comparison of graphbased ranking algorithms for publications and authors. In *Journal for Systems and Software*, 2006.

[34] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Ins Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM '07)*, 2007.

[35] Ian Soboroff, Arjen P. de Vries, and Nick Craswell. Overview of the TREC-2006 enterprise track. In *Proceedings of the 15th Text REtrieval Conference (TREC 2007)*, 2007.

[36] Daria Sorokina, Rich Caruana, and Mirek Riedewald. Additive groves of regression trees. In *Proceedings of the 18th European Conference on Machine Learning (ECML '07)*, 2007.

[37] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[38] Ellen Voorhees. The trec-8 question answering track report. In *Proceedings of Text Retrieval Conference (TREC 1999)*, 1999.

[39] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM conference on Research and development in information retrieval (SIGIR '07)*, 2007.

[40] Jun Xu, Tie yan Liu, Min Lu, Hang Li, and Wei ying Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31th annual international ACM conference on Research and development in information retrieval (SIGIR '08)*. ACM Press, 2008.

[41] Zi Yang, Jie Tang, Bo Wang, Jingyi Guo, Juanzi Li, and Songcan Chen. Expert2bole: From expert finding to bole search. In *Proceedings of the 15th ACM Conference on Knowledge Discovery and Data Mining (KDD '09)*, 2009.

[42] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector

method for optimizing average precision. In *Proceedings of the 30th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR '07)*, 2007.

[43] Chun-Ting Zhang. The e-index, complementing the h-index for excess citations. *Public Library of Science One*, 4:5, 2009.

[44] J. Zhu, S. Song, S. Rüger, and J. Huang. Modeling document features for expert finding. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008.

[45] Jianhan Zhu, Dawei Song, and Stefan Rüger. The open university at trec 2006 enterprise track expert search task. In *Proceedings of the15th Text REtrieval Conference (TREC 2006)*, 2007.