INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Learning to Rank Academic Experts

Catarina Alexandra Pinto Moreira

Dissertation for the achievement of the degree

Master in Information Systems and Computer Engineering

Committee

Chairman:	Prof. Doutor Alberto Rodrigues da Silva
Advisor:	Prof. Doutor Bruno Martins
Co-Advisor:	Prof. Doutor Pável Calado
Observer:	Prof. Doutor João Paulo Neto

November 2011

In memory of Nuno Lourenço

8th June of 2008

Acknowledgments

I would like to thank my advisors Prof. Bruno Martins and Prof. Pável Calado for their valuable help and commitment to this work. Their wise comments and suggestions contributed a lot for the development of this thesis. I also want to thank them for giving me the opportunity of participating in the European Digital Mathematics Libraries project, EuDML.

I must also thank Prof. Andreas Wichert for all the support and motivation that he has given me throughout this work. Our conversations about art, science and philosophy showed me the wonderful world of research and how rewarding it can be.

I would also like to thank Prof. Luísa Coheur for her support in a very difficult time of my life. Without her support and kindness, I do not know if I would ever be sitting here writing this Master Thesis. She has been like a mother to me and I do not know how I can ever repay her for all the goodness that she has given me. I thank her from the bottom of my heart for helping me and never giving up on me. To you Luísa, I give you an enormous thank you!

This work is dedicated to the memory of Nuno Lourenço. If I had a chance, I would like to tell him that he has given me the happiest days of my life. Thanks to him, reality has become better than a dream. He has filled an empty void in my heart and I can never thank him enough for loving me. I am a better person because of him. This work is for you Nuno. Everything that I have accomplished with this research work is in your memory. Every single word that I have written is my effort to not let you die in vain and make you remembered forever. As long as I live, I will always keep an image of you in my heart. Thank you so much for everything you have done for me. For the wonderful world that you have shown me. Farewell my love, I will never forget you...

October 2011

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 thesis 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. More specifically, the algorithms tested were Additive Groves, AdaRank, Coordinate Ascent, RankBoost, RankNet, SVMrank and SVMmap.

This thesis 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. The algorithms tested were CombSUM, CombMNZ, CombANZ, Borda Fuse, Reciprocal Rank Fuse and Condorcet Fusion. Experiments made over a dataset of academic publications for the Computer Science domain attest for the adequacy of the proposed approaches.

Keywords: Learning to Rank, Expert Finding, Information Retrieval, Neural Networks, Support Vector Machines, Boosting, Additive Models, Rank Aggregation, Data Fusion

Resumo

A tarefa de procura de peritos tem recebido imenso interesse na literatura de recuperação de informação. No entanto, o actual estado da arte ainda carece de boas abordagens para combinar diferentes fontes de evidência de uma forma óptima. Esta tese explora o uso de métodos baseados em *learning to rank* como uma boa abordagem para combinar múltiplos estimadores de relevância numa maneira óptima. Estes estimadores são derivados de conteúdos textuais, de estruturas em grafo representando padrões de citação na comunidade de peritos e de informações baseadas no perfil dos peritos. Especificamente, fizeram-se experiências com diversos algoritmos supervisionados de learning to rank representativos das abordagens pointwise, pairwise e listwise. Os algoritmos testados foram o Additive Groves, AdaRank, Coordinate Ascent, RankBoost, RankNet, SVMrank and SVMmap.

Esta tese também explora a eficácia de abordagens baseadas na agregação de rankings combinando diferentes técnicas de fusão de dados. Foram feitas experiências com algoritmos de estado da arte, nomeadamente o CombSUM, CombMNZ, CombANZ, Borda Fuse, Reciprocal Rank Fuse e Cordorcet Fusion

As experiências foram efectuadas numa colecção de dados de publicações académicas no campo da Engenharia Informática. Os resultados obtidos confirmam a adequação de ambas as abordagens propostas.

Palavras-Chave: Learning to Rank, Procura de Peritos, Recuperação de Informação, Redes Neuronais, Support Vector Machines, Boosting, Modelos Aditivos, Agregação de Rankings, Fusão de Dados

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The automatic search for knowledgeable people in the scope of specific communities or large organizations, with basis on documents describing people's activities, is an information retrieval problem that has been receiving increasing attention (Serdyukov, 2009). Usually referred to as *expert finding*, the task involves taking a short user query as input and returning a list of people, which are related to the query topic, sorted by their level of expertise.

Early experiments in the area of expert finding started in the late 90's with the pioneering work of Mattox et al. (1999). They applied information retrieval, information extraction and collaborative filtering techniques to perform the search. The ranking of the candidates was computed by the number of mentions of the name of the candidates associated with the query terms in the documents. The search was performed over newsletters and other publications. The authors considered that a candidate was an expert if it was linked to a wide range of documents and/or to a large number of documents containing the query terms.

An expert finding task also appeared at the Text REtrieval Conference (TREC) in year of 2005, due to the increasing interest in this specific retrieval problem. TREC provided a platform where researchers could test their techniques through a scenario that included the discovery of relationships between entities in a large organizational scale. The first approaches that were tested involved either (i) creating a document for each candidate and then applying simple information retrieval techniques to rank those documents, or (ii) they involved natural language processing and information extraction technologies (Craswell et al., 2005).

1.1 Motivation

Since TREC, many effective expert finding approaches have been proposed in the literature, exploring different retrieval models and different sources of evidence for estimating the candidate's expertise. However, the current state-of-the-art still lacks in principled approaches for combining different sources of evidence in an optimal way.

In traditional information retrieval tasks such as ad-hoc 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 (Liu, 2009). The general idea 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. However, few previous works have specifically addressed the usage of learning to rank approaches in the context of expert finding problems.

With my thesis, I propose the usage of supervised learning to rank algorithms as a sound approach for addressing the expert finding task, through the combination of a large pool of estimators which characterize the knowledge of an expert in an optimal way.

1.2 Thesis Proposal

With this work, I attempt to improve the performance of an expert finding system through the usage of learning to rank methods, specifically combining a large pool of estimators for expertise. A set of estimators were derived from the textual similarities between documents and queries, from the graph-structure with the citation patterns for the community of experts and from profile information about the experts. A prototype has been developed in order to validate a supervised learning approach for expert finding, where several state of the art learning to rank algorithms were tested and compared between each other.

Since relevance judgements for the task of expert finding are very hard to find, I also experimented with a rank aggregation framework in order to combine the multiple estimators of expertise. The prototype also supported several data fusion algorithms proposed in the information retrieval literature.

1.3 Contributions

The main contributions of this work can be summarized as follows.

- **A Set of Features to Measure Expertise.**
  In this thesis, it was developed a set of features which enable the characterization of the knowledge of an expert based on different sources of evidence. It was defined three different sets of features, namely text similarity features, author profile information features and features based on authorship and co-citation graphs. The textual similarity features enable the modulation of the candidate's knowledge in a directly way through the usage of traditional information retrieval techniques which measure the co-occurrences between the query topics and the documents associated with a candidate. The profile information features measure the total publication record of a candidate throughout his career, under the assumption that highly prolific candidates are more likely to be considered experts. Finally, the graph features enable the modulation of more accurate ranking models through the usage of link-based structures. These structures allow the computation of the impact of a candidate in the scientific community through citation and co-authorship links. A set of academic indexes based in the scientometrics community were also defined to determine such impact. These sets of features have already been published as part of some research works by Moreira, Martins and Calado (2011) and Moreira, Calado and Martins (2011).

- **A Supervised Learning to Rank Approach for Expert Finding.**
  This thesis explored the usage of learning to rank methods 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 with the citation patterns for the community of experts, and from profile information about the experts. These estimates correspond to the first contribution mentioned in this work. For combining the multiple sources of expertise, this thesis uses representative algorithms of the learning to rank for information retrieval literature, such as the SVMrank and the SVMmap algorithms which were respectively proposed by Joachims (2006) and Yue et al. (2007), as well as more recent algorithms such as Additive Groves proposed by Sorokina et al. (2007). Part of this experiment has been published by Moreira, Calado and Martins (2011).

- **A Rank Aggregation Approach for Expert Finding.**
  The rank aggregation methods received an increasing attention in this thesis, because the relevant judgements required by supervised learning to rank methods are very hard to find for the task of expert finding. Just like in the previous contribution, this thesis also explored

the usage of rank aggregation methods, specifically combining a large pool of estimators for expertise. These estimates correspond to the first contribution mentioned in this work. For combining the multiple sources of expertise, this work uses representative data fusion algorithms of the information retrieval literature, such as the CombSUM and the Condorcet Fusion algorithms respectively proposed by Fox and Shaw (1994) and Montague and Aslam (2002). Part of this experiment has already been published by Moreira, Martins and Calado (2011).

1.4 Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2 presents the most important concepts which are critical for understanding this work. It presents the classic models used in information retrieval, as well as recent advancements such as linked-based approaches. It also presents the most popular metrics for evaluating information retrieval systems. Finally, a brief description about statistical tests for learning and classification problems is presented.

Chapter 3 segments and describes the most relevant works of expert finding in the literature. It presents an exhaustive description of the four main approaches for expert finding, namely the candidate-based, the document-based, the hybrid and the graph-based approaches.

Chapter 4 describes three sets of features used to estimate the expertise level of a candidate given a topic, namely the textual features which are based in ad-hoc IR systems, the profile information features which are based in the publication record of an author and the graph features which correspond to relevance estimates computed from the author's co-authorship and co-citation graphs.

Chapter 5 presents the learning to rank framework for information retrieval, as well as the most relevant works in the literature using this supervised approach. It also explores the problem of expert finding as a supervised information retrieval problem under the assumption that learning to rank can be a principled approach for combining the multiple estimators of expertise, described in Chapter 4, in an optimal way.

Chapter 6 presents the proposed unsupervised rank aggregation framework for the task of expert finding, as well as the most relevant works in the literature using similar approaches. Finally, it details a set of data fusion algorithms which were used to combine the multiple estimators of expertise described in Chapter 4.

Chapter 7 describes the datasets used in the experiments, as well as the validation methodology used in the different experiments and discusses the results obtained in each one of them.

Finally, Chapter 8 presents the main conclusions of this work and provides a brief overview of all contributions of this thesis. It also presents a discussion of future directions that this thesis can point to.

Chapter 2

Concepts

This chapter presents important concepts which are critical for understanding this work. It gives a brief explanation of what is information retrieval and introduces the task of expert finding. It also presents the classic models used in information retrieval, as well as recent advancements such as linked-based approaches. It also presents the most popular metrics for evaluating information retrieval systems. Finally, a brief description about statistical tests for learning and classification problems is presented.

2.1 Information Retrieval and Expert Search

Information Retrieval (IR) generally aims at the representation, organization, storage and retrieval of documents, in order to provide the user with easy access to required information (Baeza-Yates and Ribeiro-Neto, 1999). To satisfy an information need, an IR system must be able to check the documents in a collection and rank them according to their degree of relevance to a user query expressing the information need. The major difficulty faced by an IR system resides in the computation of these relevance estimates.

Expertise retrieval is a subcategory of information retrieval where the goal is to identify a list of expert candidates who are knowledgeable about a given expertise area, by uncovering associations between those candidates and topics discussed over a document collection (Balog et al., 2007). A user starts by formulating a query with a topic of his interest. Then, the retrieval system ranks the candidates according to the expertise area expressed in the query, using available documentary evidence and profile information related to the candidate experts (Serdyukov, 2009).

In order to work, an expert search system has two requirements, namely (i) a list of candidates that can be retrieved by the system, and (ii) textual evidence on the expertise of the candidates, given in the form of e-mail messages, academic publications, blog posts, web pages, forum posts, etc. (Macdonald and Ounis, 2006).

Expert search is much more difficult than document retrieval, because (i) the expertise areas of a candidate are rare and hard to quantify and (ii) the experience of the candidates may vary (Maybury, 2006). According to Macdonald et al. (2008), the three major factors which affect the retrieval performance of an expert search system are (i) the selection of the documents associated with the candidates, (ii) the measurement of the relevance for each document which contains the query topics, and (iii) the combination of the expertise evidence from the associated documents.

A standard retrieval system cannot solve the expert search problem directly, since it would find experts strictly by ranking documents (Petkova and Croft, 2006). The system may start by retrieving documents, but it must then extract and process the textual information to uncover the associations between documents and topics, and between documents and candidates. There are two main approaches for modelling these associations. The first one is based on gathering all the information from the different candidates in profile documents, and then ranking the candidates with basis on the associations found between the topic and the query. The other approach first collects all documents that are associated with the topic and then tries to find associations to the candidate experts. In both these approaches, the expert finding system has to discover documents related to a person and then estimate the probability of that person being an expert on the query topics with basis on the textual contents. Section 3 of this document provides an exhaustive survey on the subject of expert finding.

2.2 Classical Models of Information Retrieval

In the context of adhoc document retrieval, several conceptual models have been used to represent documents and queries. The classical models of information retrieval can be classified into three major categories, namely Vector Space Models, Discriminative Probabilistic Models and Generative Probabilistic Models (i.e., Language Models).

2.2.1 Vector Space Models

In this case, documents are represented as vectors in an n-dimensional vector space, where n corresponds to the number of unique terms in the document collection, and where each term of the vector can be weighted according to its relative importance. Each document i is expressed as $D_i = (d_{i1}, d_{i2}, ..., d_{in})$, where d_{ij} corresponds to the weight of the d_{jth} term in the document i.

Having both documents and queries represented in an n-dimensional space, one possible and very effective way to measure the similarity between them is through vector matching operations such as the cosine similarity. This operation is used to measure the cosine of the angle between the vectors. The similarity results are then used to rank the documents according to their estimated relevance to the query (Salton et al., 1975).

In order to score the documents correctly, according to their importance to the query, it is critical to know which weights should be applied to each unique term of the vectors. Two of the most effective term weighting functions are the TF.IDF and the Okapi BM25 approaches. Okapi BM25 was originally proposed as a scoring function which determines the similarity of two documents (i.e, it offers a complete retrieval model). However, Okapi BM25 can also be used as a term weighting function to be combined with the cosine similarity.

TF.IDF is based on calculating the relative frequency of a word in a specific document, compared to the inverse proportion of that word over the entire document corpus (Ramos, 2001). The TF.IDF formula is given by the formula bellow, where $f_{t,d}$ is the number of occurrences of term t in document d, $|D|$ is the size of the document collection and $f_{t,D}$ corresponds to the number of documents in the collection where term t occurs.

$$TF.IDF = f_{t,d} \times \log \frac{|D|}{f_{t,D}} \tag{2.1}$$

According to Manning (2008), the TF.IDF values are high when the term t occurs many times in a small set of documents and are low when the term t occurs a few times in the document or when t occurs very often in a large number of documents (e.g., terms like articles and prepositions).

The Okapi BM25 term weighting and document-scoring function (Robertson and Zaragoza, 2009) is a combined function composed of several simpler scores with different components and parameters. The components which are involved in BM25 are the *inverse document frequency* and term *term frequency*, which have been explained in the context of the TF.IDF metric, together with the *document length* and the *average document length* of the collection. The BM25 document-scoring function is given by Equation 2.2, 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.

$$BM25(q,d) = \sum_{i \in Terms(q)} \log\left(\frac{N - Freq(i) + 0.5}{Freq(i) + 0.5}\right) \times \frac{(k_1 + 1) \times \frac{Freq(i,d)}{|d|}}{\frac{Freq(i,d)}{|d|} + k_1 \times (1 - b + b \times \frac{|d|}{\mathcal{A}})} \tag{2.2}$$

The function in Equation 2.2 measures the similarity of a query q towards a given document d in a collection by summing the weights of all the terms of the query. The results can be used to rank documents according to their relevance to the query. If we remove the $\sum_{i \in Terms(q)}$ summation from Equation 2.2 and make the formula independent of a query parameter, then Okapi BM25 can be used as a term scoring function.

One limitation of Okapi BM25 is that it does not provide any guidelines on how to choose the internal parameters k_1 and b. However, according to Robertson and Zaragoza (2009), these parameters provide good results when their values are between $0.5 < b < 0.8$ and $1.2 < k_1 < 2$.

2.2.2 Discriminative Probabilistic Models

In the case of discriminative probabilistic models, the retrieval system tries to estimate the probability that a specific document d_m is to be judged relevant or not relevant with respect to a user query q_k, i.e, $P(R|q_k, d_m)$. This is often referred to as the probability ranking principle, stating that the ranking of the documents is given by their estimated probability of relevance with respect to the query $P(R = 1|d, q)$. To estimate this probability, it is assumed that terms are distributed differently within relevant and non-relevant documents. Let $T = \{t_1, ..., t_n\}$ denote the set of terms in the collection. The set of terms which occur in document d_m^T can be represented as a binary vector $x = (x_1, ..., x_n)$ where $x_i = 1$ if $t_i \in d_m^T$ and $x_i = 0$ otherwise (Fuhr, 1992). Instead of these binary vectors, weighting functions such as TF.IDF can also be used to improve the retrieval mechanism. The probability $P\left(R|q_k, x\right)$ is estimated through a discriminative model and different documents containing the same set of terms are to be ranked with the same probability. These estimates are important to evaluate how terms in a document contribute to the relevance judgement. The documents are then ordered by decreasing estimated probability of relevance.

The original and still most influential discriminative probabilistic retrieval model is the *binary independence model* (BIM). The Binary Independence Model uses the Naive Bayes probabilistic theories and it is based on the independence assumption, i.e. it assumes that terms occur in the documents independently. Although this independence approach is not correct, studies have shown that in practice it usually gives good results (Manning, 2008). The probability that a given document is to be judge relevant is given by:

$$P(R|x,q) = \frac{P(x|R,q).P(R|q)}{P(x|q)} \tag{2.3}$$

In the formula, $P(R = 1|x, q)$ and $P(R = 0|x, q)$ are the probabilities of retrieving a relevant or non-relevant document, respectively.

2.2.3 Language Models

In the case of Language Models, a document is a good match for a query if a probabilistic generative model for the documents is capable of generating the query, which happens when the document contains the terms of the query more often.

Compared to the previously introduced discriminative probabilistic models, instead of modelling the probability of relevance of a document d for some query q, i.e. $P(R = 1|q_{k1}, d_m)$, the Language Model approach builds a model θ_d from each document d and thereafter ranks the documents based on the probability of the document model having generated the query, i.e. $P\left(q|\theta_d\right)$.

A document model can generate a query through various methods, including finite automata, n-grams or even by using probabilistic estimates based on the Bayes theorem (Manning, 2008), where the probability of a document is interpreted as the likelihood of having query q being produced from the document d.

In order to construct a document model, a document d is represented as the probability distribution of the vocabulary terms over the document, i.e. $P(t|d)$. The *maximum likelihood* estimate of a term, which is given by its relative frequency in the document, provides the simplest method for inferring an empirical document model. Considering a common approach, where the document model θ_d is a *unigram* language model, the probability of a query q is the product of the individual term probabilities, and is given by Equation 2.4.

$$P(q|\theta_d) = \prod_{t \in q} p(t|\theta_d)^{n(t,q)} \tag{2.4}$$

In the above equation, $n(t, q)$ corresponds to the number of times term t occurs in the query q. However, this approach has the limitation of giving a zero probability when one or more query terms do not appear in the document, and therefore smoothing techniques need to be applied (Balog, 2008). A common smoothing technique in language models is to use a parameter λ which assigns some weight to all documents in the collection, even if a given query term does not appear in the document. This parameter assumes values between 0 and 1.

A known approach to build a document model using smoothing techniques is given by the Jelinek-Mercer smoothing method, which is given by Equation 2.5.

$$P(q|\theta_d) = \prod_{t \in q} (1 - \lambda_t) P(t|d) + \lambda_t P(t) \tag{2.5}$$

The book by Manning (2008) provides additional information on language models for IR.

2.3 Link-Based Information Retrieval

The first generations of search engines used textual contents exclusively to generate ranked lists of results. However, applying these methods to the Web did not provide good results. When considering the Web, too many documents may contain the query terms and it is hard to discriminate which among them are in fact the most relevant. Modern Web search engines use linkage data together with the document's textual contents. They are able to gather information mined from the documents themselves, as well as information from the linkage structure of the Web (Sebastiani, 2003). This linkage structure is composed of hypertext links from source documents to target documents, and it can be seen as a citation network.

More formally, a citation network is a directed graph composed of links between source nodes (citing) to target nodes (cited). This can be useful to determine the relevance of a document, because each time a document (e.g. an academic article or a web page) is cited, it reflects its quality/significance/impact to the author of the document making the citation. The links between documents are called $inLinks$ and are hypertext links pointing to a web page. The area of link-based information retrieval borrowed techniques from bibliometrics (i.e., the study of citation patterns in academic publications) and developed approaches for estimating document importance and similarity between documents, using linkage information.

One of the best known linkage analysis techniques currently used in information retrieval is PageRank, an algorithm proposed for Google which assumes that the prestige of a node is proportional to the sum of the PageRank scores from the nodes that link to it. In addition, a node (i.e., a document) that is connected to nodes with a high PageRank value also receives a high PageRank score. Equation 2.6 shows how to compute the PageRank (P_r) score recursively for a node i:

$$P_r(i) = \frac{0.5}{N} + 0.5 \sum_{j \in inLinks(L,i)} \frac{\alpha_j P_r(j)}{outLinks(L,j)} \tag{2.6}$$

In the formula, N is the number of nodes in the graph, $\alpha = 1, 2, ...L$ is the weight of each node and L represents the direct links between a source node and a target node. There are many extensions of the PageRank algorithm which will be addressed in detail over Section 4.3.

To measure the similarity of two nodes in a graph (e.g. two documents or two authors), it is often useful to take into account the citation patterns between the elements of the collection. Two popular similarity metrics based on citation networks are bibliographic coupling and co-citation.

In Bibliographic Coupling, two documents are similar if they share some references and the strength of that similarity is measured by the number of common references (Kessler, 1963). Figure 2.1 shows a directed graph illustrating the concept of bibliographic coupling. In this figure, documents A and B have a bibliographic coupling of 3.

Co-Citation is a variation of Bibliographic-Coupling originally proposed by Small (1973). Two documents d_1 and d_2 are similar if there are other documents citing both d_1 and d_2. The strength of their similarity depends on the frequency with which the two documents are cited together in a same document. Figure 2.1 shows a citation graph illustrating the co-citation approach. In the figure, documents A and B have a co-citation score of 3.

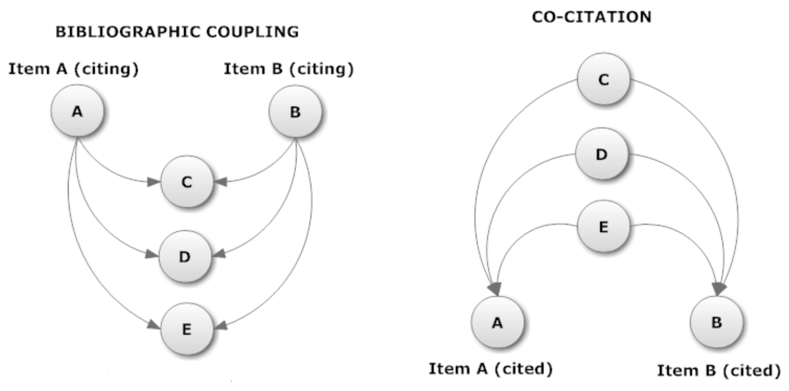

Figure 2.1: Bibliographic Coupling and Co-Citation (Garfield, 2001)

2.4 Evaluation in Information Retrieval

The evaluation of IR systems is typically made through laboratory experiments relying on a document collection. In the field, this is known as the Cranfield methodology, since the first experiments that used this method were made at the Cranfield Institute of Technology in the late 1950. According to Rowe et al. (2010), the components of this methodology are:

- A collection of documents to be searched (i.e., the test collection);

- A series of queries answered by some documents in this collection, representing possible information needs of the users;
- The results of the IR system when trying to match the information needs with the document collection (i.e., the retrieved documents);
- Performance measures based on relevance judgements, stating which documents in the collection are relevant for each query.

Early experiments in the field were based on a complete relevance assessment of the document collection. Later experiments on the context of the Text Retrieval Conference (TREC) used much larger document collections, using a *pooling* methodology instead of complete relevant assessments. The TREC methodology is very robust and avoids the exhaustive assessment of all documents by the users (Belew, 2000). In the pooling method, the results submitted by the participants are used to form a pool of documents for each topic, by collecting the highly ranked documents from all submissions (Gozalo et al., 2000). This guarantees that the list of documents assessed for relevance is as comprehensive as possible (Sormunen, 2002). If a document is not included in the pool, then it remains unjudged and is assumed irrelevant (Belew, 2000). In order to enable evaluation, a binary scale is often used to determine the relevance of the documents. TREC states that one can only make binary judgements (i.e., determining whether a document is relevant) if there is one piece of supporting information in the document (no matter how small it is) that is relevant to the query (Sormunen, 2002).

The two simplest measures of effectiveness, based on binary relevance judgements, are Precision and Recall. These two measures are defined for the simple case where an IR system returns an unordered set of documents for a query.

Precision measures the fraction of the returned results which are relevant to the query and is given by the following equation:

$$Precision = \frac{|\{RelevantDocuments\} \bigcap \{RetrievedDocuments\}|}{|RetrievedDocuments|} \quad (2.7)$$

Recall measures the fraction of the relevant documents in the collection which were returned by the system and is given by:

$$Recall = \frac{|\{RelevantDocuments\} \bigcap \{RetrievedDocuments\}|}{|RelevantDocuments|} \quad (2.8)$$

In a ranked retrieval context, appropriate sets of retrieved documents should be given in the top k retrieved documents. In order to discriminate between ranked lists, it is necessary to extend these basic measurements or create new ones (Manning, 2008). Four other measures which have into account the top k retrieved documents are the precision at rank k, the mean average precision, the normalized discount cumulative gain and the mean reciprocal rank.

Precision at rank k is used when a user wishes only to look at the first k retrieved documents. The precision is calculated at that rank position through Equation 2.9.

$$P@k = \frac{r\left(k\right)}{k} \tag{2.9}$$

In the formula, $r(k)$ is the number of relevant documents retrieved in the top k results.

The Mean Average Precision (MAP) directly considers the order in which relevant documents are returned by a system. For some query, the Average Precision is the average of precision values obtained in the top k documents returned by a system. The MAP is nothing more than the mean value of the Average Precision, computed for various queries (Manning, 2008)

$$MAP = \frac{1}{|Q|}\sum_{j=1}^{|Q|} AP(j) \tag{2.10}$$

In the above formula, AP is the Average Precision, which is given by:

$$AP = \sum_{i=1}^{k} \frac{Precision(i)rel[i]}{NumOfRelevantInstances} \tag{2.11}$$

In Equation 2.11, $rel\left[i\right]$ is a function which assumes the value 1 if $rel\left[i\right] = TRUE$ (i.e., if the document at rank position i is relevant) and 0 otherwise.

Precision at rank k and Mean Average Precision are both based on binary judgements. The Normalized Discount Cumulative Gain (NDCG) metric, on the other hand, is based on non binary judgements. It not only gives more importance to the top k retrieved documents, but also takes into account how these k documents are ordered. NDCG measures the degree of relevance of documents based on their position in the result list produced by the system for some query. It assumes that highly relevant documents are more valuable than marginally relevant ones, and the greater the ranked position of a relevant document is, the less valuable it is for the user, because the more likely it is that the user will never examine the document (Järvelin and Kekäläinen, 2002).

In short, NDCG is used to emphasize the highly relevant documents which appear on top of the list of results returned by a query. It is given by Equation 2.12.

$$NDCG_p = Z_p \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2\left(1 + i\right)} \tag{2.12}$$

In the formula, rel_i is the relevance score assigned to document i and Z_p is a normalization factor calculated by normalizing the set of the top retrieved documents, guaranteeing that a perfect ranking's NDCG at position n equals 1 (Manning, 2008).

The Mean Reciprocal Rank (MRR) is a measure used to evaluate the correctness of one relevant document in a list of possible responses to a query. In Equation 2.13, for a query q, the rank position of its first relevant document is denoted as $rank_i$ and the Reciprocal Rank is given by the multiplicative inverse of the rank of the first correct answer, that is $\frac{1}{rank_i}$. It follows that the Mean Reciprocal Rank is the mean value of the Reciprocal Ranks of the results for a set of queries (Sanderson, 2010).

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i} \tag{2.13}$$

For further information about these evaluation metrics, the reader is referred to Manning (2008).

2.5 Evaluation with Significance Tests

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 or not.

In the literature, Dietterich (1998) made an overview of five of the most relevant significance tests which can be applied for comparing supervised classification learning algorithms. These tests are the McNemar's test, the test for the difference of two proportions, the resampled t test, the cross validated t test and the $5x2cv$ test.

- **The McNemar's test.** It is based on a χ^2 test which makes a comparison between the number of examples expected in the null hypothesis and the number of observed examples.

- **Test for the Difference of Two Proportions.** It is based on a normal distribution, measuring the difference between the error rates of two different classification algorithms.

- **The Resampled t test.** It is based on a series of trials, where the sample data is randomly divided into a training set and a testing set. Two different classification algorithms, A and B, are applied in those sets. The significance performance is given by a Student's t test applied to the difference between the proportion of misclassified examples returned by A and the misclassified examples returned by B.

- **The k-fold Cross Validated paired t test.** It works just like the above method, but instead of creating the training and testing sets randomly, the cross validated t test method randomly divides the samples into k disjoint sets of equal size. In each trial the test set consists in one of the k sets and the training set corresponds to the remaining $k-1$ sets.

- **The $5x2cv$ Test.** This method corresponds to performing the above method, for $k = 2$, five times. For each time this method is executed, the data sample is randomly split into two equal sets. Given two different learning algorithms, each one of them is trained on each set, and tested on the other. This, returns four error estimates, and by performing the difference between them, it ends up with two error estimates. These are used to allow the computation of a t distribution.

2.6 Summary

This chapter presented the most important concepts which are critical for understanding this work. It gave a brief explanation of what is information retrieval and introduced the task of expert finding. It also presented the classic models used in information retrieval, as well as recent advancements such as linked-based approaches. It also presented the most popular metrics for evaluating information retrieval systems. Finally, a brief description about statistical tests for learning and classification problems was presented.

Chapter 3

Related Work

Although expert search is a recent concern in the information retrieval community, there are already many research efforts addressing this specific task. In this chapter, the most relevant works in this area are described. After a careful analysis of the related literature, I suggest to classify the different approaches to the expert search problem into four categories, namely (i) candidate-based, (ii) document-based, (iii) hybrid and (iv) graph-based approaches. Learning to rank and rank aggregation approaches can be used to combine features derived from models based on the other types, and they will be described in Chapters 5 and 6 respectively. The four basic classes of expert search approaches will be detailed in the following Sections.

3.1 Candidate-Based Approaches for Expert Finding

In candidate-based approaches, the system gathers all textual information about a candidate and merges it into a single document (i.e., the profile document). The associations between each document in the collection and a candidate are uncovered through the occurrence of personal identifiers of the candidate experts, such as names or email addresses, in the original documents. The profile document can be seen as the representation of the candidate's knowledge and is ranked by the probability of the candidate given the query topics. This approach assumes that the document and the candidate are conditionally independent, and that the more a candidate talks about some topic, the bigger are the chances of him being an expert in that topic.

One of the first expert finding systems following this approach was proposed by Craswell et al. (2001). The system was based on a standard web search engine and the ranking was determined by text similarity measures computed between the query and the expertise evidence from the profile documents associated to each candidate expert.

Another well known candidate-based model was formalized by Balog et al. (2006). Their approach offers a general probabilistic framework for modelling the expert finding task, which can be extended in order to provide stronger associations between candidates and topic terms. This approach is usually referred to as *Model 1* and it uses Language Models to rank candidates according to the probability of the respective candidate model, obtained from the profile document, generating the query topics. Figure 3.2 shows the general framework of this model.

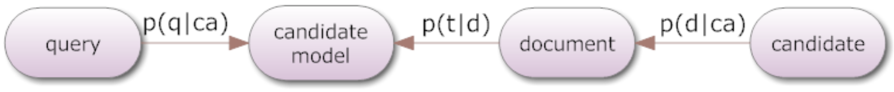

Figure 3.2: Candidate-Based Approach (based on Balog (2008))

Petkova and Croft (2006) presented a general approach for representing the knowledge of a candidate expert as a mixture of language models from associated documents. Instead of merging all evidences of a candidate into one single profile document, they formed a group with the set of documents which are associated with a candidate, and weighted the contribution of each document independently. The candidates were ranked with adhoc retrieval methods. Although one may think that this approach is very similar to the well known document-based approach, it has some differences that put it in the category of candidate-based approaches. In document-based expert finding, the system first gathers all documents which contain the query topics and only then it extracts the candidate names from that set. What Petkova and Croft (2006) proposed was a system that first gathers all documents which are associated with some candidate. With that document set, they generated a ranked list of candidates according to the query topics.

Later, Petkova and Croft (2007) introduced the idea of dependency between experts and document terms. They formalized a candidate-centred document representation which uses positional information (i.e., the distance between a candidate name and the query terms) to weight the strength of the associations between terms and candidates. Specifically, they used proximity kernels to account with this distance, by fitting a multinomial density function around the occurrences of a candidate in each given document.

Balog et al. (2009) also explored the positional information and formalized a candidate-based document representation which includes a window surrounding the candidate's name in each document. The idea behind this method is that the closer a candidate is to a term, the more likely is that term to provide evidence of his knowledge.

In order to improve retrieval performance in adhoc retrieval tasks, query expansion methods (a.k.a. pseudo-relevance feedback) are also often used. These techniques can also be applied in the context of expert finding. The main idea of query expansion consists in examining the top retrieved documents (i.e., the pseudo-relevance set) by an IR system, afterwards using the specific information of these documents in order to re-weight the query terms and consequently improve the ranking of retrieved documents (Salton and Buckley, 1997).

Macdonald and Ounis (2007*b*) employed two *Divergence From Randomness* techniques to extract informative terms from the pseudo-relevance set. They also showed that query expansion techniques can be used in the context of a voting approach for the expert finding (Macdonald and Ounis, 2006). They proposed the candidate centric query expansion approach, where the pseudo-relevance set is taken from the final ranking of candidates generated by a query. One can see this model as the pseudo relevance feedback extension for the candidate-based approach. However, they showed that the candidate centric query expansion method performed poorly when compared to the document centric one.

Later, when investigating the poor results from the candidate centric approach, Macdonald and Ounis (2007*a*) discovered that it suffered from topic drift. Since a candidate can have several unrelated expertise areas, when performing pseudo-relevance feedback, those expertise areas will gain relevance and will therefore lead to wrong results. To overcome the problem, the authors proposed to use a measure of cohesion which measures how strongly related a query topic is to a candidate, this way obtaining better results.

Serdyukov et al. (2007) proposed to apply query expansion through the usage of pseudo-relevance feedback, specifically through language models obtained from the top retrieved documents and the top retrieved candidates. This approach takes into account other expertise areas related to a candidate expert without suffering from topic drift.

Fang and Zhai (2007) applied the probabilistic ranking principle to develop a general framework from which the candidate-based and document-based models for expert finding could be derived. They also showed how query expansion techniques, such as the association of different weights to each candidate representation and the topic expansion in order to give more information terms to the original query, can improve the performance of the models from the framework.

Balog and de Rijke (2008*b*) tried to extend the candidate-based model by adding non-local evidence, i.e., expertise evidence obtained using information that is not available in the immediate proximity of a candidate name in a document. The idea is that the mere co-occurrence of a person with a query term is not necessarily an indication of expertise of that person on the topic. If a candidate is associated with a large number of documents, independently of the query terms, then the probability of that candidate being associated to some particular document should be low, because it does not contribute significantly as evidence. In the same way, if a candidate is associated with many documents which are semantically similar, and then is associated with another document which is semantically different from the others, then the probability of that later document being associated with the candidate should also be low.

3.2 Document-Based Approaches for Expert Finding

In document-based approaches, the system starts by gathering all documents which contain the expertise topic terms included in the query. Afterwards, the system uncovers which candidates are associated with each of those documents.

Contrary to the candidate-based approaches that were previously presented, which represented the candidate's knowledge with a single profile document, the document-based approach is based on adhoc information retrieval methods. First, the system gathers all documents which are relevant to the information need. Then, for each relevant document, the system determines the probabilities of the candidates being associated with it. Finally, the system adds up the individual relevance probabilities of each candidate and returns the list of possible candidate experts. The idea behind this approach is that if there is a candidate name in a piece of text where the query topics occur often, then it is probable that the candidate is an expert in that topic.

This approach was first used by Cao et al. (2006) in the Text REtrieval Conference of 2005. They proposed a two-stage language model where the first stage, called the relevance model, determines whether a document is relevant to the query topics or not. The second stage, called the co-occurrence model, determines whether or not a query topic is associated with a candidate. To improve this model, they added co-occurrence sub-models which give more weight to a document when a query and a person occur within the same window of text. For an extreme case, this window can have the size of the entire document. They also applied sub-models which give different weights according to the positions of the document in which a candidate could appear.

A well known document-based model was formalized by Balog et al. (2006). This approach is commonly referred to as *Model 2* and language models are used to rank the candidates according to the probability of a document model having generated the query topics. Figure 3.3 shows the general framework of this document-based model.

Balog et al. (2009) also explored the usage of positional information and formalized a document representation which includes a window surrounding the candidate's name. The idea behind this approach is that the closer a candidate name is to a term, the more likely for that term to

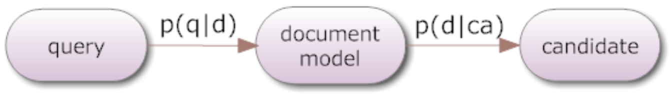

Figure 3.3: Document-Based Approach (based on (Balog, 2008))

contribute as evidence of his knowledge.

Balog et al. (2009) exhaustively tested the candidate-based and document-based approaches. They concluded that the document-based approach outperforms the other one. They also tested these approaches together with the usage of positional information and, this time, the candidate-based approach achieved better results.

Query expansion techniques were also used in the context of the document-based approaches for the expert finding task. Macdonald and Ounis (2007*b*) proposed the document centric query expansion approach, where after ranking the documents relatively to a query, a pseudo-relevance set is taken from the top ranked documents of the document ranking. *Divergence From Randomness* techniques are then used in order to extract informative terms from the pseudo-relevance set composed of the top-retrieved documents.

Macdonald and Ounis (2008) formalized a voting framework combined with data fusion techniques to be applied in the context of expert finding. In this framework, the system first ranks all documents respectively to the query topics. Then, each candidate associated with the top retrieved documents receives an implicit vote which states the expertise relevance of the candidate relatively to the query. The ranking of each candidate is given by the aggregation of the votes of each document. The authors tested their framework with 12 different voting techniques, where each of the voters represented sources of evidence that can be derived from the top retrieved documents. These voting techniques can be seen as approaches for aggregating scores. This approach achieved better results than the approaches based on language models.

In later work, Macdonald et al. (2008) applied clustering techniques to the candidate's profile documents to identify the main interests of each candidate, under the assumption that the main expertise areas will be given by the largest clusters. Only the profiles of candidates who had a number of associated documents bigger than a given threshold were clustered and they used the cosine of the angle between the average vector of each cluster as the clustering distance. The clusters were ranked by the number of documents that they contained and the top clusters were chosen as representatives of each candidate's expertise. The authors combined these clusters with a voting approach, assuming that votes for candidates retrieved by documents belonging to the largest clusters were given more weight and, consequently, the retrieval performance of the system would improve. They also included heuristics based on URL length and the number of incoming links to identify candidate homepages, because they contain personalized and professional information about the candidate.

3.3 Hybrid Approaches for Expert Finding

In the case of hybrid approaches, the expert finding system combines the ideas of the candidate and document-based approaches. This means that the system starts by ranking the candidate experts through their own language models, which were generated from the top retrieved documents according to the query topics. Then, the level of expertise on the search topic is given by the frequency of the candidate's mentions in the top ranked documents.

The most important model which falls into this category, as far as I know, was formalized by Pavel Serdyukov in his PhD thesis (Serdyukov, 2009). He proposed a *Person-Centric model* which does not follow the independence assumption used in the candidate and document-based approaches. Instead, this model assumes that candidates are responsible for generating the terms within retrieved documents, which means that there should be a dependency relation between the query terms and a candidate.

The idea behind this model is that if a candidate is mentioned in a document, then he might be responsible for its content, either explicitly as the author of the document or implicitly as a recipient. This leads to the fact that a candidate expert is a strong indicator for a document topic. Figure 3.4 shows the general framework of this model.

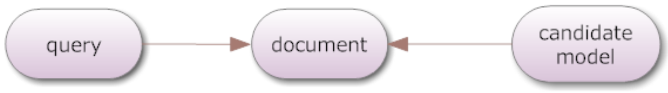

Figure 3.4: Person-Centric Approach

The person-centric model starts just like a document-based approach, by retrieving the top ranked documents according to the query topics. Since these documents do not have a language model associated to them, the system requests the candidate experts to generate the document terms using their own language models. Then, it ranks candidates just like in the candidate-based approach, by combining (i) the probability of generation of the query by the candidate's language model, and (ii) the prior probability of being an expert, expressed in terms of a candidate's activity in the top retrieved documents. To build the candidate language models, two methods based on the *Expectation-Maximization* algorithm were proposed, namely one that considered that the probability of the association between a candidate and a document was fixed and fully dependent on the field of a document where the candidate appeared, and another that obtained those probabilities dynamically by predicting the contribution of specific people to a document with basis on their intermediately estimated language models (Serdyukov, 2009). Experimental results attested for adequacy of this model, showing that it can outperform candidate-based and document-based approaches.

3.4 Graph-Based Approaches for Expert Finding

The approaches presented so far do not take into account the linkage structure between candidates and documents. They are only focused in uncovering associations between topics and candidates by analysing the documents' textual content which a candidate is related to. Hence, they miss information by not considering the expertise of directly and indirectly linked candidates, and the relevance of documents which are in a near neighbourhood of a candidate.

Using graph-based approaches, the system starts by ranking documents according to the query topics. From this set of documents, the system then extracts the contained candidates into another set. In this model, the documents and the candidates can be seen as nodes and their containment relations can be seen as directed edges. Together, they form an *expertise graph*.

Campbell et al. (2003) developed an expert search task approach for email documents, arguing that these documents are a valuable source of expertise. The approach first collected all emails which contained the query topics. Then, this set of retrieved documents enabled the creation of a directed social graph from email corpora, using people as nodes and the email headers *from/to* as edges. Finally, they applied a modified version of the HITS algorithm (Kleinberg, 1999) that only considered authority scores to rank the candidates according to their expertise level.

Zhang et al. (2007) applied expert search approaches in a web forum. Since each thread in a forum discusses a specific topic, they formalized these threads as *post/reply* directed social graphs, where each participant was a node and the edges corresponded to directed links associating a user posting a question to a user replying to it. They also applied network-ranking algorithms like PageRank and HITS, concluding that the characteristics of the network structure matter. For instance, in the forum on which they worked on, people replying to many users are not necessarily experts, because they could be answering only to *newbies*. They compared their results against human judgements and they concluded that using PageRank and HITS provided better results than using a standard document-based approach.

Chen et al. (2006) made a similar study, where they also built social networks based on *post/reply* headers from forum corpora. They ranked the nodes in the network with PageRank and HITS. Instead of using an online community forum like Zhang et al. (2007), they used the W3C corpus of the TREC 2006 enterprise track. They concluded that, for this dataset, both PageRank and HITS did not performed very well when compared with a standard document-based approach.

Instead of analysing community based forums or email corpora, Liu et al. (2005) focused on finding authoritative scientists in digital libraries. They modelled a co-authorship network where each node is a person and each edge links a person to another if they have worked together. They applied several of the measures that are traditionally applied in undirected graphs, namely degree, closeness, betweenness, PageRank, and others. According to Liu et al. (2005), the degree measures how many connections tie authors to their immediate neighbours in the network. Closeness focuses on how close an author is to all other authors. Betweenness is based on determining how often a particular node is found on the shortest path between any pair of nodes of the network. The authors also defined a new measure based on PageRank, namely the *Author Rank*, to estimate the impact of an individual author in the network. The conclusion that they have taken is that these measures perform well in the identification of important authors. However, these co-authorship networks do not take into account the level of expertise of the authors, because they ignore the relations which are only in the context of a certain topic.

Serdyukov et al. (2008) represented candidate experts and documents in expertise graphs and modelled the principle of expert finding by three types of probabilistic random walks, namely finite, infinite and absorbing. To model the graph, they first applied a document-based approach where the system retrieves the most relevant documents which contain the query topics. Then, from this set of relevant documents, the contained candidates are extracted. Candidates and documents become nodes in the expertise graph and their containment relations become directed edges. Then, they exploited all known connections in the graph by including further links, such as expert-to-expert, making the graph very dense and enabling a better propagation of relevance, consequently enabling people to receive expertise evidence from documents, even if these documents are not in an immediate proximity. Finally, they separately tested the three random walk methods and concluded that expertise graphs with finite random walks provide much better results than infinite or absorbing random walks.

More recently, Deng et al. (2011) 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. In their approach, they precomputed the authority scores just like in an ordinary AuthorRank algorithm, but at query time, they used language models to first retrieve the top communities (i.e. conferences or journals where authors publish their papers) which are related with the query topics. Finally, their query sensitive AuthorRank computed the probability of an author being an expert by taking into consideration the probability of a community given the query topics and the probability of an author being associated to that community. This query sensitive AuthorRank achieved better results than the representative state of the art algorithms proposed by (Balog et al., 2006), namely Model 1 and Model 2.

3.5 Comparison Between the Previously Presented Approaches

As already mentioned, TREC provided a platform for researchers to test their expert finding approaches on. Table 3.1 shows the top three best-performing approaches in the 2005-2008 editions of the TREC Enterprise Track. Analysing Table 3.1, one can see that the TREC-2005

TREC Edition	Top 3 Approaches	MAP	MRR
2005	1st Fu et al. (2006)	0.2749	0.7268
2005	2nd Cao et al. (2006)	0.2688	0.6244
2005	3rd Yao et al. (2006)	0.2174	0.6068
2006	1st Zhu et al. (2007)	0.6431	0.9609
2006	2nd Bao et al. (2007)	0.5947	0.9358
2006	3rd You et al. (2007)	0.5639	0.9043
2007	1st Zhu et al. (2008*a*)	0.4632	0.6333
2007	2nd Duan et al. (2008)	0.4427	0.6131
2007	3rd Zhu et al. (2008*b*)	0.4337	0.5802
2008	1st Balog and de Rijke (2008*a*)	0.4490	0.8721
2008	2nd Shen et al. (2008)	0.4214	0.7241
2008	3rd He et al. (2008)	0.4126	0.7611

Table 3.1: Retrieval performance of the top 3 best systems in TREC

edition was the one which had the lowest results. This is understandable, because it was the first time that an expert search competition was held. Most of the 2005 competitors used adhoc information retrieval methods to find ways of estimating the candidate's expertise from the dataset. As explained in the beginning of this work, expert finding is more difficult than an adhoc document retrieval task, and traditional information retrieval methods are not enough.

The 2006 edition of TREC had the best results of all editions. Expert finding started to receive plenty attention from researchers, and many different approaches based on language models were developed. Some of the main approaches for expert finding were first shown in this edition, such as the candidate-based and the document-based approaches. These new formalizations were found to be very good and very effective when compared to the basic adhoc information retrieval methods presented in TREC-2005.

In the editions of 2007 and 2008, the results decreased when compared to the 2006 edition. In 2007, TREC changed the dataset to the publicly available pages of an Australian organization called CSIRO. TREC-2007/2008 participants were provided with only a structural template of email addresses used by CSIRO employees. Most participants had to get around spam protection, check if similarly looking addresses belonged to the same employee, and filtering non-personal addresses (Serdyukov, 2009). This dataset may have turned the expert finding task a little more difficult and, for that reason, the results were not as good as in the 2006 edition.

Some of the approaches mentioned in the previous sections, although not having achieved one of the top three places in TREC, are very well formalized and are easy to understand. We also have many of the models from the winning teams are just more effective extensions of the base models described in the previous sections. A comparative analysis of the previous systems is also shown in Table 3.2, with basis on the results obtained in TREC.

	2005		2006		2007	
State of the Art Approaches	MAP	MRR	MAP	MRR	MAP	MRR
Petkova and Croft (2006)	0.2850	0.6496	-	-	-	-
Balog et al. (2006)	0.1894	0.2434	-	-	-	-
Chen et al. (2006)	-	-	0.4949	-	-	-
Petkova and Croft (2007)	-	-	0.6193	0.9541	-	-
Fang and Zhai (2007)	0.2040	-	0.4650	-	-	-
Macdonald and Ounis (2007*b*)	0.2231	-	0.5689	-	-	-
Macdonald and Ounis (2007*a*)	0.2271	-	0.5783	-	-	-
Macdonald and Ounis (2008)	**0.2917**	-	**0.6571**	-	-	-
Macdonald et al. (2008)	0.2324	-	0.5657	-	0.4319	0.5742
Balog and de Rijke (2008*b*)	-	-	-	-	**0.5465**	0.2880
Serdyukov et al. (2008)	-	-	0.4130	0.8070	0.4070	0.5660
Balog et al. (2009)	0.2725	**0.6800**	0.4697	**0.9558**	0.4633	**0.6236**
Serdyukov (2009)	0.1235	0.4700	0.4125	0.8120	-	-

Table 3.2: Retrieval performance of the approaches described in Chapter 3

3.6 Summary

In this chapter, the most relevant works in area of expert finding were described. The related literature was segmented into four categories, namely (i) candidate-based, (ii) document-based, (iii) hybrid and (iv) graph-based approaches.

In the candidate-based approach, the system gathers all textual information about a candidate and merges it into a single document. The associations between each document in the collection and a candidate are uncovered through the occurrence of personal identifiers of the candidate experts in the original documents. The profile document can be seen as the representation of the candidate's knowledge and is ranked by the probability of the candidate given the query topics.

The document-based approaches are based on adhoc information retrieval methods where the system first gathers all documents which are relevant to the information need and then, for each relevant document, the system determines the probabilities of the candidates being associated to the document. Finally, the system adds up the individual relevance probabilities of each candidate and returns the list of possible candidate experts.

In the hybrid approaches, the system combines the ideas of the candidate and document-based approaches. This means that the system starts by ranking the candidate experts through their own language models, which were generated from the top retrieved documents, for the query topics. Then, the level of expertise on the search topic is given by the frequency of the candidate's mentions in the top ranked documents.

In the graph-based approaches, the system starts by ranking documents according to the query topics. From this set of documents, the contained candidates are extracted into another set. The documents and the candidates can be seen as nodes and their containment relations can be seen as directed edges. The usage of link based algorithms such as PageRank and Hits are used in order to measure the candidate's relevance towards a topic.

Remembering Table 3.2 one can see that the Voting Model proposed by Macdonald and Ounis (2008) outperformed all other approaches proposed. This leads to the conclusion that this rank aggregation approach based on data fusion techniques has a bigger impact in the task of expert finding, than all the approaches based in generative models. This is a very interesting fact, since in this work it is attempted to develop a rank aggregation prototype also based in data fusion techniques.

Chapter 4

Features to Estimate Expertise

In order to retrieve relevant candidates in an expert finding task, one requires a collection of textual documents which describes the degree of knowledge of a candidate. This knowledge can be represented by a set of features. The definition of these features must be very carefully chosen, because machine learning approaches, as well as rank aggregation approaches, require a set of features which are able to identify whether or not a candidate is relevant given a specific query topic. If these features are not able to represent such relevance, then the learning machine will not be able to learn a ranking model capable of making a good classification and therefore, the whole learning to rank principle collapses.

In this chapter, it is described a set of features to estimate the expertise level of a candidate given a query topic. These features fall into three categories, namely textual features, author's profile information features and graph features. The textual features are similar to the ones used in adhoc IR systems (e.g., TF.IDF and BM25 scores). The author's profile information features are based on telling how important an author is in the scientific community, by looking at their publication record along the years. Finally, the graph features correspond to the importance and relevance estimates computed from the author's co-authorship and co-citation graphs.

4.1 Text Similarity Features

Similarly to previous document-based approaches for expert finding, we also considered the usage of textual similarities between the contents of the documents associated with an author and the query topics in order to build estimates of expertise. The hypothesis behind these features is that if there is a piece of text associated with an author where the query topics occur often, then it is probable that the author is an expert in that topic. In the scope of academic digital libraries, the associations between documents and experts can be easily obtained from the authorship information. For each query-expert pair, the set of considered textual features is as follows:

The Query 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 4.14 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 d_j.

$$TF_{q,a} = \sum_{j \in Docs(a)} \sum_{i \in Terms(q)} \frac{Freq(i, d_j)}{|d_j|} \tag{4.14}$$

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 4.15. 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.

$$IDF_q = \sum_{i \in Terms(q)} \log \frac{|D|}{f_{i,D}} \tag{4.15}$$

The Document Length (DL), corresponding to the total number of words in the documents associated with the author. The DL feature is query independent, so an author has always the same DL value, no matter the query.

The Number of Unique Authors (NUA) in documents, corresponding to the total number of unique authors which are associated with documents containing at least one of the query terms. Since the NUA feature is author independent, all the authors for a given query have the same NUA value.

The Aggregated Okapi BM25, corresponding to the sum of the BM25 scores over all the documents associated with the author, for all the individual terms in the query. Equation 4.16 presents this BM25 formula, where $Terms(q)$ is the set of terms from query q, $Docs(a)$ corresponds to the documents having a as author, the number of occurrences of term i in document d_j is given by $Freq(i, d_j)$, $|d_j|$ corresponds to the number of terms in document d_j and A represents 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, since most previous IR experiments from the literature use these default values for the k_1 and b parameters.

$$BM25_{q,a} = \sum_{j \in Docs(a)} \sum_{i \in Terms(q)} \log \left(\frac{N - Freq(i) + 0.5}{Freq(i) + 0.5} \right) \times \frac{(k_1 + 1) \times \frac{Freq(i,d_j)}{|d_j|}}{\frac{Freq(i,d_j)}{|d_j|} + k_1 \times (1 - b + b \times \frac{|d_j|}{\mathcal{A}})} \tag{4.16}$$

The Averaged Okapi BM25 corresponding to the mean value of the BM25 scores computed over all the documents associated with the author, for all the individual terms in the query. The BM25 formula is given by Equation 4.16 and, in the computation of this feature, one simple replaces the summation over all documents $j \in Docs(a)$ by an average over these values.

The Maximum Okapi BM25, corresponding to the maximum value of the BM25 scores computed all over the documents associated with the author, for all individual terms in the query. The BM25 formula is given by Equation 4.16 and, in the computation of this feature, one simple replaces the summation over all documents $j \in Docs(a)$ by the maximum of these values.

The Aggregated Jaccard Coefficient corresponding to the sum of the Jaccard Coefficient scores over all the documents associated with an author, for all the terms of the query. Equation 4.17 presents the Jaccard Coefficient formula, where $Terms(q)$ is the set of terms from query q, $Docs(a)$ corresponds to the documents having a as author, $|Terms(q) \cap d_j|$ corresponds to the number of intersections between the query terms and the terms of the j_{th} document associated with the author a and $|Terms(q) \cup d_j|$ corresponds to the total amount of terms in the query and the j_{th} document.

$$J(q,a) = \sum_{j \in Docs(a)} \frac{|Terms(q) \cap d_j|}{|Terms(q) \cup d_j|} \tag{4.17}$$

The Average Jaccard Coefficient corresponding to the mean value of the Jaccard scores computed over all the documents associated with the author, for all the terms in the query. The Jaccard formula is given by Equation 4.17 and, in the computation of this feature, one simple replaces the summation over all documents $j \in Docs(a)$ by an average over these values.

The Maximum Jaccard Coefficient, corresponding to the maximum value of the Jaccard scores computed all over the documents associated with the author, for all terms in the query. The Jaccard formula is given by Equation 4.17 and, in the computation of this feature, one simple replaces the summation over all documents $j \in Docs(a)$ by the maximum of these values.

The Aggregated/ Average/ Maximum Okapi BM25 scores of the conferences and journals where an author publishes. Since most of the times the authors do not use directly the query terms in their publications' titles or abstracts, it is assumed that an author who publishes a lot in conferences or journals containing the query topics is an expert in that topic. Thus, we compute BM25 scores similarly to those corresponding to Equation 4.16, but replacing the documents associated to author a, $Docs(a)$, by the conferences and journals where the author a has published his work, $Conf(a) \cup Journ(a)$.

In the computation of the 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 abstract of the articles.

4.2 Features Based on Profile Information

Since the goal of expert search is to find knowledgeable people in certain topics, features based on profile information about the candidates play an important role. These profile features are related to the amount of publications that an author has made throughout his career. The hypothesis behind these features is that highly prolific authors are more likely to be 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 is as follows:

The Total Number of Publications of an author, with and without the query topics, considering that highly prolific authors should be more relevant.

The Total Number of Journal Publications of an author, with and without the query topics, considering that publications on journals are of highly quality and therefore very important to estimate relevance.

Years passed since the author's last publication containing the query terms. It is considered that authors who published more recently on the subject of the query are more likely to be considered experts than others who published long ago.

Years passed since the author's first publication containing the query terms. It is considered that the first authors publishing on the query subject are more likely to be experts.

The Time Interval in years, spanning through the dates of publications associated with the oldest and the newest articles of the author, under the assumption that authors with a longer career are more likely to be considered experts.

The Average Number of Publications per Year of an author, under the assumption that authors who maintain a regular publication activity should be considered more relevant. This feature is computed by considering the interval of years from the oldest to the latest publication associated with the candidate expert.

The Average Number of Journal Publications per Year associated with an author, under the assumption that authors who maintain a regular journal publication activity are more likely to be considered relevant candidates.

4.3 Features Based on Connections between Experts

When considering academic digital libraries, linkage structures such as co-citation and co-authorship graphs are usually used in order to discriminate documents and improve ranking scores. These graph-based structures can offer effective approaches for estimating the importance of the contributions of particular publications, publication venues or individual authors. The features that were considered under this category are the following:

The Total Number of Citations for the Papers associated with an author which contain the query terms. It is considered that highly cited authors are more likely to be considered experts on the query topics.

The Average Number of Citations for the papers associated with the author which contain the query terms. It is considered that authors who are cited very often are more likely to be experts in the subjects expressed in the query terms.

The Average Number of Citations, per year, over all papers belonging to an author that contain the query terms. It is considered that authors who are cited very often are more likely to be considered experts in the subject expressed in the query topics.

The Maximum Number of Citations for an individual paper associated with an author that contains the query terms. It is considered that authors of at least one highly influential paper are more likely to be experts.

The Total Number of Unique Collaborators who participated with the author in publications which contain the query terms. It is considered that authors who collaborate with many different people are more likely to be considered experts.

The Hirsch Index of the author. This index measures both scientific productivity and the apparent scientific impact of a candidate expert (Hirsch, 2005). An author 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. It is considered that authors with a high Hirsch index are more likely to be considered experts.

The Hirsch-b Index, which is an extension of the Hirsch index for evaluating scientific topics in general (Banks, 2006). 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.

The Contemporary Hirsch Index of the author, which is an extension of the Hirsch index that adds weights to each cited article with respect to its age. This means that older articles are given less weight than more recent ones (Sidiropoulos et al., 2007). 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 follows:

$$S^c(i) = \gamma * (Year(now) - Year(i) + 1)^{-\delta} * |CitationsTo(i)| \tag{4.18}$$

According to Sidiropoulos et al. (2007), the parameters γ and δ should be set to 4 and 1 respectively, in order to get better results. These parameters mean that the citations for an article published during the current year count 4 times, the citations for an article published 4 years ago count only one time, the citations for an article published 6 years ago count $4/6$ times, and so on.

The Trend Hirsch Index of the author, which is used to estimate the impact of a researcher's work in a particular time instance by assigning to each citation an exponentially decaying weight according to the age of the citation (Sidiropoulos et al., 2007). 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:

$$S^t(i) = \gamma * \sum_{\forall x \in C(i)} (Year(now) - Year(x) + 1)^{-\delta} \tag{4.19}$$

According to Sidiropoulos et al. (2007), the parameters γ and δ should be set to 4 and 1 respectively, in order to provide good results.

The Individual Hirsch Index of the author, which is used to reduce the co-authorship effects of influential authors who contribute to the Hirsch Index of the author. This feature is computed by dividing the value of the standard Hirsch Index by the average number of authors in the articles that have a contribution to the Hirsch Index of the author (Batista et al., 2006).

The a-Index of the author, which measures the magnitude of his most influential articles. For instance, if an author has a Hirsch Index of h that has a total of $N_{c,tot}$ citations towards his papers, then he has an a-index of $a = N_{c,tot}/h^2$.

The g-Index of the author, also quantifying scientific productivity with basis on his publication record (Egghe, 2006). Given a set of articles associated with the author, 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 Hirsch's Index of the institution of the author, under the assumption that authors from high impact institutions are more likely to be considered experts in the topics expressed in the query. An institution has a Hirsch Index of h if h of its N_p papers have at least h citations each, and the other $(N_p - h)$ papers have at most h citations each.

The a-index of the institution of the author, which estimates the impact of an institution by measuring the magnitude of its most influential articles. For instance, if an institution has a Hirsch Index of h and has a total of $N_{c,tot}$ citations towards its papers, then it has an a-index of $a = N_{c,tot}/h^2$.

The g-index of the institution of the author, also under the assumption that authors from high impact institutions are more likely to be considered experts. Given the set of articles associated with the institution, ranked in decreasing order of the number of citations that they received in the past, the g-index is the (unique) largest number such that the top g articles received on average at least a number of g^2 citations.

Besides the above features, the ones that were defined in the previous work of Chen et al. (2007) were also taken into consideration. In their work, Chen et al. (2007) considered a set of network features to estimate the influence of individual authors through the use of PageRank algorithm, which was already described in Section 2.3. The following set of features, based on PageRank, were considered for estimating candidate expertise:

The Sum of PageRank Values associated to papers of the author which contain the query terms. Each citation link in the graph is given a score of $1/N$, where N is the number of authors in the paper. PageRank is afterwards computed on this weighted graph, representing citations between papers. The idea is that authors with a high PageRank score are more likely to be considered experts in the topics expressed in the query.

The Average of PageRank Values associated to papers of the author which contain the query terms. Each citation link in the graph is given a score of $1/N$, where N is the number of authors in the paper. The average of the PageRank is afterwards computed on this weighted graph, representing citations between papers.

The PageRank Value of an Author Computed Over a Directed Graph where each link represents the number of citations between authors. To each citation link between two authors a_1 and a_2, this approach gives a weight corresponding to the number of papers where author a_1 has cited author a_2.

The PageRank Value of an Author Computed over an Undirected Graph which represents collaborations between authors. In order to compute PageRank, the undirected graph will be first converted to a directed one by placing bi-directional links between each pair of nodes that are connected. To each citation link between two authors a_1 and a_2, this approach gives a weight corresponding to the number of papers where author a_1 collaborated with author a_2.

4.4 Summary

This chapter described the set set of features considered for estimating the expertise level of a candidate given a topic. These features fall into three categories, namely textual features, author profile features and graph features. The textual similarity features enable the modulation of the candidate's knowledge in a direct way through the usage of traditional information retrieval techniques which measure the co-occurrences between the query topics and the documents associated with a candidate (e.g., TF.IDF and BM25 scores). The author's profile information features are based on telling how important an author is in the scientific community, by looking at their publication record along the years. They measure the total publication record of a candidate throughout his career, under the assumption that highly prolific candidates are more likely to be considered experts. Finally, the graph features enable the modulation of more accurate ranking models through the usage of link-based structures which allow the computation of the impact of a candidate in the scientific community. A set of academic indexes based in the scientometrics community were defined in order to determine such impact.

It is reminded that the definition of these features must be very carefully chosen, because machine learning approaches require a set of features which are able to identify whether or not a candidate is relevant given a specific query topic. If these features are not able to represent such relevance, then the learning machine will not be able to learn a ranking model capable of making a good classification and therefore, the whole learning to rank principle collapses.

Chapter 5

Supervised Learning to Rank

Information retrieval essentially deals with ranking problems. An adhoc retrieval system checks the documents of a collection and ranks them according to their degree of relevance to a user query expressing the information need. The major difficulty faced by an IR system resides in the computation of these relevance estimates. The previous section presented a set of approaches which required the manual tuning of parameters (e.g., the k_1 and b in BM25 and the α in PageRank). In order to get a good ranking performance, these parameters need to be tuned for the validation set. And that is not a trivial task. For that reason, there was an increasing growth of interest in automatic ranking systems.

In this chapter, it is described a supervised learning to rank framework which is able to automatically tune parameters and build ranking models, through the usage of hand labelled training data (e.g., document collections containing relevance judgements for specific sets of queries), this way leveraging on data to combine different estimators of relevance, proposed in the previous chapter, in an optimal way.

In this chapter, it is introduced the learning to rank framework for information retrieval, as well as a brief description of the representative algorithms in the literature. Some state of the art approaches, which explore the problem of expert finding as a supervised learning problem, are also presented. Finally, the learning to rank approach developed for the purpose of this thesis is detailed.

5.1 The Supervised Learning to Rank Framework

Learning to rank for information retrieval is a particular approach to the traditional information retrieval ranking problems. It automatically tunes parameters and builds ranking models through the usage of hand labelled training data (e.g., document collections containing relevance judgements for specific sets of queries), this way leveraging on data to combine different estimators of relevance in an optimal way. The learned raking model can then sort documents according to their degree of relevance to a given query. In the scope of expert finding, learning to rank (L2R) methods can automatically help to sort candidate experts according to the query topics.

Figure 5.5 provides an illustration of the general approach which is commonly used in most learning to rank methods. It consists in two separate steps, namely training and testing.

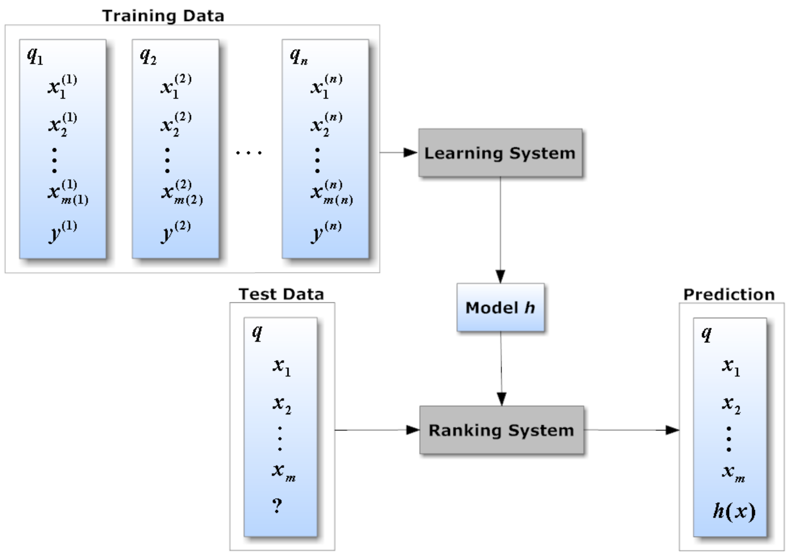

Figure 5.5: The general Learning to Rank framework (Liu, 2009)

In this framework, $q_i(i = 1, ..., n)$ corresponds to the set of n queries for the training step, $x^{(i)} = \{x_j^{(i)}\}_{j=1}^{m(i)}$, with m being the number of documents associated with query q_i, are the feature vectors associated to each query and $y^i(i = 1, ..., n)$ are the corresponding set of relevance judgements. When applying a specific learning method to this training set, the system combines the different estimators of relevance in an optimal way, learning the corresponding ranking model. During the learning process, a loss function is applied to measure the inconsistencies between the hypothesis space h and the ground truth label y.

In the test step, the learned ranking model is applied to a new given query in order to sort documents according to their relevance to the information need. The ranked document set is then returned as a response to the query.

Liu (2009) classified the existing learning to rank algorithms into three major approaches, namely the *pointwise*, *pairwise* and *listwise* approaches.

5.1.1 The Pointwise L2R Approach

In pointwise approaches, since the relevance degrees can be regarded as numerical or ordinal scores, the learning to rank problem is seen as either a regression, classification or ordinal regression problem. The idea behind the pointwise approach is that given feature vectors of each single document of the training data for the input space, the relevance degree of each of those documents can be predicted with scoring functions, and throughout these scores one can sort all documents and produce the final ranked list. A loss function is applied in order to measure the individual inconsistencies between the scoring function and the ground truth labels. Many supervised machine learning algorithms for regression, classification or ordinal regression can be readily used for this purpose.

In regression based algorithms, the relevance degree is seen as a real number. The two most representative algorithms which are based in regression are the Polynomial Regression Function (Fuhr, 1989) and the Subset Ranking with Regression (Cossock and Zhang, 2006).

In classification based algorithms, the ground truth label is not regarded as a quantitative value. Representative approaches are the Discriminative Model for IR (Nallapati, 2004) and the Multi-Class Classification for Ranking (McRank), an algorithm proposed by Li et al. (2008).

In ordinal based algorithms, the learning model considers the ordinal relationship of the ground truth labels. The aim of the ordinal regression problem is the use of thresholds in order to define a scoring function, discriminating the outputs of the scoring function into different ordered categories (Liu, 2009). The most representative algorithms of this approach are the Perceptron Based Ranking (PRanking), algorithm proposed by Crammer and Singer (2001), and the Additive Groves algorithm Sorokina et al. (2007).

The advantage of the pointwise approach is that it directly supports the application of existing theories and algorithms on regression or classification. However, the information order between documents cannot be considered in the training step, because the algorithm receives single documents as input. Since evaluation measures in IR are based in two fundamental criteria, namely query-level and position-based data, this loss of document order information is a major problem, because the evaluation measures which take into account the position of the documents will not be directly optimized and, consequently, the pointwise approach may unconsciously overemphasize unimportant documents.

5.1.2 The Pairwise L2R Approach

In pairwise approaches, since the relevance degree can be regarded as a binary value which tells which document ordering is better for a given pair of documents, learning to rank methods are seen as a classification problem. The general idea behind the pairwise approach is that given feature vectors of pairs of documents of the training 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. The loss function used in the pairwise approach takes into consideration the relative order of each pair of documents. Several different pairwise methods have been proposed in the literature. The most representative methods are RankNet (Burges et al., 2005), RankBoost (Freund et al., 2003) and SVMrank (Joachims, 2002).

Notice that the classification problem that this approach addresses is not the same as the one addressed by the pointwise approach. In the pointwise approach, the learning method operates on every single document. On the other hand, in the pairwise approach, the learning method operates on every two documents under investigation.

The advantage of the pairwise approach is that it directly supports the application of existing theories and algorithms on classification. However, the loss function only takes into account the relative order of the pair of documents and therefore it is very difficult to derive the order of the documents in the final ranked list.

5.1.3 The Listwise L2R Approach

In listwise approaches, the learning to rank problem takes into account an entire set of documents associated with a query as instances, and trains a ranking function through the minimization of a listwise loss function defined on the predicted list and the ground truth list.

The idea behind the listwise approach is that, given feature vectors of a list of documents of the training 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. The loss function used in the listwise approach takes into consideration the positions of the documents in the ranked list of all the documents associated with the same query (Liu, 2009). This is a difficult optimization problem, because most evaluation measures are not continuous functions with respect to the ranking model's parameters. Continuous approximations or bounds on evaluation measures have been successfully used in this task.

Several different listwise methods have been proposed in the literature. The most representative ones are SoftRank (Taylor et al., 2008), SVMmap (Yue et al., 2007), AdaRank (Xu and Li, 2007), ListNet (Cao et al., 2007) and Coordinate Ascent (Metzler and Croft, 2007).

The major advantage that the listwise approach has over the pointwise and the pairwise approaches is that its loss function takes into account the positions of the documents in a ranked list of all documents associated with the query. Since evaluation measures in IR consider the document positions, this approach generally improves the performance.

5.2 Existing Learning to Rank Approaches for Expert Finding

As described in Chapter 3, many works on expert finding have already been conducted. However, most of the proposed methods are based on language models and linkage information, with very few studies having attempted to use learning to rank approaches.

The most related approach to the system implemented in the context of this MSc thesis is given by Yang et al. (2009), who proposed an expert search tool which employs learning to rank techniques to learn a function to rank candidate experts. Their framework consisted of three major modules, namely the *data preparation*, the *expert finding* and the *bólè search* modules. In the data preparation module, they extracted academic data from structured datasets, such as the DBLP Computer Science Bibliography Dataset[1], and from unstructured web pages (e.g., the researchers homepages). The extracted and integrated information was stored in an academic network database. In the expert finding module, the authors used SVMrank as the supervised learning algorithm to rank candidates. Finally, they extended the expert finding system in order to perform a bólè search, that is, trying to identify the best supervisors in a specific filed. In the bólè search module, the authors tried to explore the advantages of the generic labelled training data for expert finding. They defined features which were based in language models and in the expertise scores of a researcher, such as the number of publications of a researcher. They evaluated the system using human judgements and obtained better results in comparison to the generic expert finding approaches based on languages models.

Other interesting works in the domain of expert search which use supervised learning techniques belong to Macdonald and Ounis (2011) and to Fang et al. (2010).

Macdonald and Ounis (2011) proposed a learning to rank approach which uses rank aggregation methods. They created a feature generator composed of three components, namely a document ranking model, a cutoff value and rank aggregation methods. The ranking model consists in adhoc information retrieval methods such as BM25 and TF-IDF. The cutoff works as follows.

[1]`http://dblp.uni-trier.de/`

Suppose that we have BM25 as a ranking model, then the cuttoff would select the top N documents returned by BM25. The rank aggregation methods used consist in data fusion techniques which are applied to the top N documents. After the features have been generated, Macdonald and Ounis (2011) made experiments with two listwise learning to rank algorithms, namely AdaRank and Metzlet's Automatic Feature Selection Algorithm (AFS). The authors concluded that the AdaRank algorithm performed better for finding experts in an organizational task, however the AFS algorithm outperformed AdaRank when looking for experts in blogs.

Fang et al. (2010) proposed a learning framework for expert search based on probabilistic discriminative models. They defined a standard logistic function which enabled the integration of various sets of features in a very simple model. One should notice that the integration of features, in the state of the art proposed language models, is a difficult task because they require extra modelling assumptions which increase the complexity of the model. The authors also defined query-expert pairs by two feature vectors (i) one which verifies if a document is relevant for a query topic (document-based approach) and (ii) another which verifies if an expert is relevant to a document (candidate-based approach). They also proposed a set of features to be contained in those feature vectors. Their features included, for instance, standard language models, document features (ex. title containing query topics), proximity features, etc. In their work, Fang et al. (2010) showed that their approach outperformed the existing state of the art methods based on language models.

5.3 The Learning to Rank Prototype for Expert Finding

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 relevance 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, BM25, etc.) to link-based features computed from networks encoding relations between the experts in E (e.g., PageRank). These

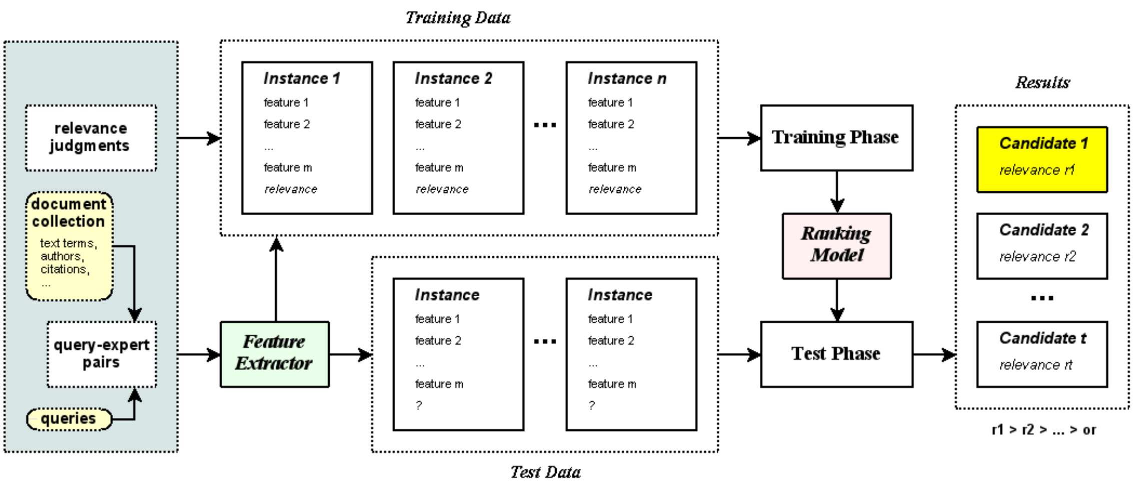

Figure 5.6: The learning to rank prototype

features have been specified in Chapter 4. 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. Figure 5.6 provides a general illustration of the developed prototype.

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.

In this thesis, it was 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 Rank-Boost. And Finally, the pointwise approach tested was the Additive Groves algorithm. The following subsections present a full description of each one of these algorithms.

5.3.1 AdaRank

The AdaRank listwise method, proposed by Xu et al. (2008), 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). Algorithm 1, describes the AdaRank algorithm proposed by Xu et al. (2008).

Algorithm 1 AdaRank

```
Input: training set $S = \{(q_i, e_i, y_i)\}_{i=1}^{m}$ and parameters $E$ and $T$
Output: ranking model $h(x) = sign(\sum_{t=1}^{T} \alpha_t(r_t(x)))$
1: $P_t(i) = 1/m$, $(i = 1, ..., m)$
2: for $t = 1, ..., T$ do
3:     Create weak ranker $r_t$, with weighted distribution $P_t$ on the training set $S$
4:     Choose $\alpha_t$
           $$\alpha_t = \frac{1}{2}.\ln \frac{\sum_{i=1}^{m} P_t(i)[1 + E(\pi(q_i, e_i, r_t), y_i)]}{\sum_{i=1}^{m} P_t(i)[1 - E(\pi(q_i, e_i, r_t), y_i)]}$$
5:     Create model $h_t$
           $$h_t(\vec{x}) = \sum_{k=1}^{t} \alpha_k r_k(\vec{x})$$
6:     Update distribution $P_{t+1}$
           $$P_{t+1}(i) = \frac{\exp\{-E(\pi(q_i, e_i, h_t), y_i)\}}{\sum_{j=1}^{m} \exp\{-E(\pi(q_j, e_j, h_t), y_i)\}}$$
7: end for
```

Following Algorithm 1, for each round t, AdaRank keeps a distribution of weights over the queries in the training set. This weight is defined as P_t and $P_t(i)$ corresponds to the weight on the i_{th} training query q_i at round t. When the algorithm starts, these weights are all equally distributed over the queries. At each round, the AdaRank algorithm increases the weights of the experts which were not ranked properly by the created model h_t. As a consequence, the learning of the following round will be focused only on the creation of weak rankers that can work on the experts belonging to queries which were wrongly ranked (Xu et al., 2008).

The quality of the weak rankers is measured by the performance measure E weighted by P_t through Equation 5.20, where $\pi(q_i, e_i, r_t)$ is the permutation created for query q_i, the corresponding set of experts e_i and the ranking function r_t. Xu et al. (2008) proposed to construct the weak rankers using the features individually. That is, the features chosen as weak rankers are the ones which have the optimal weight performance among all the other features. This way, the learning process of the AdaRank algorithm consists of repeatedly selecting features and linearly combining them. When the weak ranker is built, the algorithm associates to the weak ranker a weight $\alpha_t > 0$ which measures the importance of r_t.

$$max_k \sum_{i=1}^{m} P(i).E(\pi(q_i, e_i, r_k), y_i) \tag{5.20}$$

5.3.2 Additive Groves

The Additive Groves pointwise method, introduced by Sorokina et al. (2007), 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.

Additive Groves 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. Algorithm 2, describes the Additive Groves algorithm proposed by Sorokina et al. (2007).

Algorithm 2 Classical Additive Groves

```
Input: training set $S = \{(e_i, y_i)\}_{i=1}^{m}$ and parameters $N$ and $\alpha$
Output: ranking model $h(x) = sign(\sum_{i=1}^{n} Tree_i^{\alpha_j, n})$
1: for $i = 1$ to $N$ do
2:     $Tree_i^{\alpha,N} = 0$
3: end for
4: repeat
5:     for $i = 1$ to $N$ do
6:         newTrainingSet = { e, y - $\sum_{k \neq i} Tree_k^{\alpha,N}(e)$ } ;
7:         $Tree_i^{\alpha,N}$ = trainTree($\alpha$, newTrainingSet)
8:     end for
9: until (change from the last iteration is small)
```

Following Algorithm 2, the first tree in the grove is trained using the entire original dataset, given by a set of pairs (e, y) (where e is an expert and y is its respective relevance judgement). Supposing that $\hat{T_1}$ is the model computed by the first tree, then the next tree to be trained will focus only in the examination of the prediction errors of $\hat{T_1}$, in order to build a new ranking model which is able to classify elements intractable by the previous model. The algorithm continues this way until the root of the mean squares error can no longer be improved. The final model is given by simply combining all the models computed in all trees of the grove.

Since a single grove can easily suffer from overfit when the trees start to become very large, the authors introduced the bagging procedure to the model in order to overcome overfitting. Bagging is a method which reduces a model's performance by reducing variance. On each bagging iteration, the algorithm takes a bootstrap sample from the training set, and uses it to train the full model of the grove. When repeating this procedure a number of times, the algorithm ends up with an ensemble of models. The final prediction of the ensemble on each test data point is an average of the predictions of all models. (Sorokina et al., 2007)

5.3.3 Coordinate Ascent

The Coordinate Ascent listwise method, proposed by Metzler and Croft (2007), is an optimization algorithm, used in unconstrained optimization problems, which builds a ranking model by directly maximizing an information retrieval performance measure (e.g., Mean Average Precision).

The basic idea of Coordinate Ascent is to iteratively optimize a multivariate objective function by solving a series of one dimensional searches (Metzler and Croft, 2007). 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. Equation 5.21 shows the general optimization formula.

$$\lambda_i = argmax_{\lambda_i} E(q_i, e_i, y_i) \tag{5.21}$$

In the above equation, E corresponds to the performance measure on the training data (q_i, e_i, y_i).

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.

5.3.4 RankBoost

The RankBoost pairwise method, proposed by Freund et al. (2003), 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. Algorithm 3, describes the RankBoost algorithm proposed by Freund et al. (2003).

Algorithm 3 RankBoost

```
Input: training set $S = \{(q_i, e_i, y_i)\}_{i=1}^m$ and parameters $T$ and $\theta$
Output: ranking model $h(x) = sign(\sum_{t=1}^T \alpha_t (r_t(x_0) - r_t(x_1)))$
1: $P_t(i) = \frac{1}{m \times m}$, $(i = 1, ..., m)$
2: for $t = 1, ..., T$ do
3:     Create weak ranker $r_t$, with weighted distribution $P_t$ on the training set $S$
4:     Choose $\alpha_t$
           $\alpha_t = \frac{1}{2}.\ln \frac{1 + \sum_{x_0,x_1}^m P_t(x_0,x_1)[r_t(x_0) - r_t(x_1)]}{1 - \sum_{x_0,x_1}^m P_t(x_0,x_1)[r_t(x_0) - r_t(x_1)]}$
5:     Update distribution $P_{t+1}$
           $P_{t+1}(x_0,x_1) = \frac{P_t(x_0,x_1)\exp[\alpha_t(r_t(x_0) - r_t(x_1))]}{Z_p}$, where $Z_p$ is a normalization factor
6: end for
```

Following Algorithm 3, for each round t, RankBoost keeps a distribution of weights over the queries in the training set. These weights are defined as P_t and $P_t(i)$ corresponds to the weight on the i_{th} training query q_i at round t. When the algorithm starts, these weights are all equally distributed over the queries. At each round, the RankBoost algorithm increases the weights of the ranked pairs of experts which were not ranked properly by the created model h_t. As a consequence, the learning of the following round will be focused only on the creation of weak rankers that can work on the pairs of experts belonging to queries which were wrongly ranked.

Freund et al. (2003) proposed to construct the weak rankers using the features individually in the same way as described for the AdaRank algorithm. Equation 5.22 shows the weak ranker used in RankBoost. This weak ranker function also outputs 1 or 0 if an expert e is ranked respectively above or below a threshold θ on the list of candidates returned by the query q.

$$min \sum_{m}^{x_0,x_1} P(x_0, x_1).[r_t(x_0) - r_t(x_1)] \tag{5.22}$$

5.3.5 RankNet

The RankNet pairwise method, proposed by Burges et al. (2005), 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 with respect to that weight through the usage of the backpropagation algorithm.

The RankNet algorithm receives as input the parameter $epochs$, which is the number of iterations which provide the network with an input and update the network's weights, and the parameter $hiddenNodes$, which corresponds to the number of nodes to be contained in the network's 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. 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 correctly the value of an expert never seen before. Algorithm 4, describes the RankNet algorithm proposed by Burges et al. (2005) and Jena et al. (2007).

RankNet uses as training data a set of tuples of the form $< X_i, X_j, P_{ij} >$, where X_i corresponds to the feature vector associated to expert i and X_j to the expert j. P_{ij} is the probability that expert i has a bigger rank than expert j. Equation 5.23 is a logistic function which is responsible to map the outputs of the network to the probabilities P_{ij} of the experts.

$$P_{ij} = \frac{e^{o_{ij}}}{1 + e^{o_{ij}}} \tag{5.23}$$

The cross entropy cost function adopted by RankNet is given by Equation 5.24, where $\overline{P}_{ij}$ is the target values for the posterior probabilities P_{ij} and o_{ij} corresponds to the output of the network

Algorithm 4 RankNet

```
Input: training set $S = \{(q_i, e_i, y_i)\}_{i=1}^{m}$ and parameters $epochs$ and $hiddenNodes$
Output: ranking model
 1: Initialize weight vector $W$
 2: $ep \leftarrow 0$
 3: for $ep < epochs$ do
 4:     for each tuple of experts $(X_i, X_j, P_{ij})$ taken from the training set $S$ do
 5:         Compute the output layer: $o_i$ for expert $X_i$ and $o_j$ for expert $X_j$
 6:         Compute $\frac{\delta o_i}{\delta W}$
 7:         Compute $\frac{\delta o_j}{\delta W}$
 8:         Compute $(\frac{\delta o_i}{\delta W} - \frac{\delta o_j}{\delta W})C'(o_{ij})$
 9:     end for
10:     Update weight vector $W$
            $$W \leftarrow W - \gamma \sum^{|S|} (\frac{\delta o_i}{\delta W} - \frac{\delta o_j}{\delta W})C'(o_{ij})$$
11:     Compute the total error
            $$Error \leftarrow \frac{\sum^{|S|} C_{ij}}{|S|}$$
12:     if $Error > toleranceError$ then
13:         break to avoid overfiting
14:     end if
15:     $ep \leftarrow ep + 1$
16: end for
```

and is given by $o_{ij} = f(X_i) - f(X_j)$.

$$C_{ij} = -\overline{P}_{ij} o_{ij} + \log(1 + e^{o_{ij}}) \tag{5.24}$$

In order to minimize the error function, RankNet uses a modification of the backpropagation algorithm, where the gradient of the error function is expressed just like in Equation 5.25.

$$\frac{\delta C_{ij}}{\delta W} = \frac{\delta[-P_{ij} o_{ij} + log(1 + \exp o_{ij})]}{\delta W} = (\frac{\delta o_i}{\delta W} - \frac{\delta o_j}{\delta W})C'(o_{ij}) \tag{5.25}$$

5.3.6 SVMmap

The SVMmap listwise method, introduced by Yue et al. (2007), builds a ranking model through the formalism of structured Support Vector Machines (Tsochantaridis et al., 2005), 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.

Suppose $x = \{x_j\}_{j=1}^{m}$ is the set of all the experts associated with a training query q, and suppose that $y_{u,v}^{(i)}$ represents the corresponding ground truth labels. Any incorrect label for x is represented as y^c. SVMmap solves the following quadratic optimization problem:

$$\min \frac{1}{2}||w||^2 + \frac{C}{m}\sum_{i=1}^{m}\xi^{(i)}$$

$$\text{s.t. } \forall y^{c(i)} \neq y^{(i)}, w^T\Psi(y^{(i)}, x^{(i)}) >= w^T\Psi(y^{c(i)}, x^{(i)}) + 1 - AP(y^{c(i)}) - \xi^{(i)} \tag{5.26}$$

In Equation 5.26, the margin term $\frac{1}{2}||w||^2$ controls the complexity of the ranking model w. The method introduces slack variables, $\xi_{u,v}^{(i)}$, which measure the degree of misclassification of the datum x_i. Each time there is a wrong output y, SVMmap requires a constraint. In the constraints, Ψ is called the joint feature map, whose definition is:

$$\begin{aligned} \Psi(y, x) &= \sum_{u,v:y_u=1,y_v=0}(x_u - x_v) \\ \Psi(y^c, x) &= \sum_{u,v:y_u=1,y_v=0}(x_u^c, y_v^c)(x_u - x_v) \end{aligned} \tag{5.27}$$

Since there are an exponential number of incorrect labels for the experts, it is a big challenge to directly solve the optimization problem involving an exponential number of constraints for each query. In their work, the authors suggested the usage of the cutting plane method contained in the formalism of structured SVMs to efficiently tackle this issue by maintaining a working set with those constraints with the largest violation:

$$\text{V}iolation \triangleq 1 - AP(y^c) + w^T\Psi(y^c, x) \tag{5.28}$$

5.3.7 SVMrank

The SVMrank pairwise method, introduced by Joachims (2006), 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 by Equation 5.29 over a set of n training queries $\{q_i\}_{i=1}^n$, their associated pairs of experts $(x_u^{(i)}, x_v^{(i)})$ and the corresponding relevance judgement $y_{u,v}^{(i)}$ over each pair of experts (i.e., pairwise preferences resulting from a conversion from the ordered relevance judgements over the query-expert pairs):

$$\begin{aligned} &\min \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \sum_{u,v:y_{u,v}^{(i)}} \xi_{u,v}^{(i)} \\ &\text{s.t. } w^T(x_u^{(i)} - x_v^{(i)}) >= 1 - \xi_{u,v}^{(i)} \ , \\ &\text{if } \ y_{u,v}^{(i)} = 1, \ \xi_{u,v}^{(i)} >= 0 \ , \ i = 1, \ldots, n \end{aligned} \tag{5.29}$$

Differently from standard SVMs, the loss function in SVMrank is a hinge loss defined over expert pairs. The margin term $\frac{1}{2}||w||^2$ controls the complexity of the pairwise ranking model w. This method also introduces slack variables, $\xi_{u,v}^{(i)}$, which measure the degree of misclassification of the datum x_i. The objective function is increased by a function which penalizes non-zero $\xi_{u,v}^{(i)}$, and the optimization becomes a trade off between a large margin, and a small error penalty.

Just like any SVM algorithm, SVMrank receives as input the parameter C which affects the trade-off between model complexity and the proportion of non-separable samples.

5.4 Summary

This chapter introduced the learning to rank framework for information retrieval, giving a brief description of the representative approaches from the literature. It also presented state of the art approaches which explore the problem of expert finding as a supervised learning to rank approach. Finally, this chapter presented the learning to rank approach specifically developed for the purposes of this work.

The learning to rank algorithms covered in this chapter can fall into one of three approaches, namely the pointwise approach, the pairwise approach and the listwise approach.

The pointwise approaches receive as input single candidates from the training data and predict their relevance by applying scoring functions to each individual document.

The pairwise approaches receive as input a pair of candidates from the training data and predict the relevance of those pairs through scoring functions which aim to minimize the average number of misclassified expert pairs.

Finally, the listwise approaches receive as input a set of candidates from the training data and predict their relevance by using scoring functions which try to directly optimize an information retrieval evaluation metric.

This chapter also presented a set of state of the art learning to rank algorithms. The algorithms presented were Additive Groves, RankBoost, RankNet, SVMrank, AdaRank, Coordinate Ascent and SVMmap.

The Additive Groves pointwise method builds a single small tree (i.e. a grove, from the training data). 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 RankBoost pairwise method trains one weak ranker at each round and the final model corresponds to a linear combination of weak rankers. In each iteration, RankBoost increases the weights of the misclassified expert pairs, in order to give more attention to these misclassifications in the following round.

The RankNet pairwise method uses a multilayer neural network with a modified 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 number of misclassified expert pairs by minimizing the value of the cost entropy function by adjusting each weight in the network with the backpropagation algorithm.

The SVMrank pairwise method builds a ranking model, through the formalism of support vector machines, by attempting to minimize the number of misclassified expert pairs. This is achieved by constraining the default support vector machine optimization problem to perform the minimization on each pair of experts.

The AdaRank listwise method works just like RankBoost, the difference resides in how the weak rankers are chosen. At each iteration, AdaRank increases the weight of the experts which performed poorly concerning some information retrieval evaluation metric and decreased the weights of the instances which performed very well. This way, the next iteration will only be focused in the misclassified expert, improving the classification process.

The Coordinate Ascent listwise method is an algorithm which attempts to solve unconstrained optimization problems. Coordinate ascent, iteratively, holds all features except for one, which is fixed, and tries to reoptimize the problem with respect to the fixed feature, by performing a parameter search in an one dimensional space.

Finally, the SVMmap listwise method builds a ranking model under the formalisms of support vector machines and tries to minimize a loss function which measures the difference between the performance of a perfect ranking and the minimum performance of an incorrect ranking.

Chapter 6

Rank Aggregation for Expert Finding

Chapter 4, proposed a set of features based on textual similarities, on the profile information of the authors and on relevance estimates computed over co-authorship and co-citation graphs. In this chapter, we are concerned in how to combine these estimates in order to determine accurately which are the most knowledgeable people in some topic.

The problem of combining various ranking lists for the same set of candidates, in order to obtain a more accurate and more reliable ordering, can be defined as Rank Aggregation (Dwork et al., 2001). Data fusion techniques consist in the methods used to combine these different rankings.

Montague and Aslam (2002) have experimented data fusion techniques in the domain of search engines, where they concluded that data fusion can provide significant advantages for a search engine for two major reasons. Firstly, because different retrieval methods often return very different irrelevant documents, however they always return the same relevant ones. Secondly, it provides more consistent and reliable performance than individual search engines.

In this chapter, it is argued that rank aggregation methods, based on data fusion techniques, provide significant advantages over the representative language models approaches proposed in the state of the art. We leverage on existing algorithms to build a single expert finding model which performs accurately in digital libraries. In this chapter, it is introduced the general rank aggregation framework for expert search as well as some state of the art approaches, which explore the problem of expert finding as a rank aggregation problem. Finally, representative algorithms of the information retrieval literature, which were experimented in this work, are described.

6.1 The Rank Aggregation Framework for Expert Finding

The general rank aggregation framework for expert finding is illustrated in Figure 6.7.

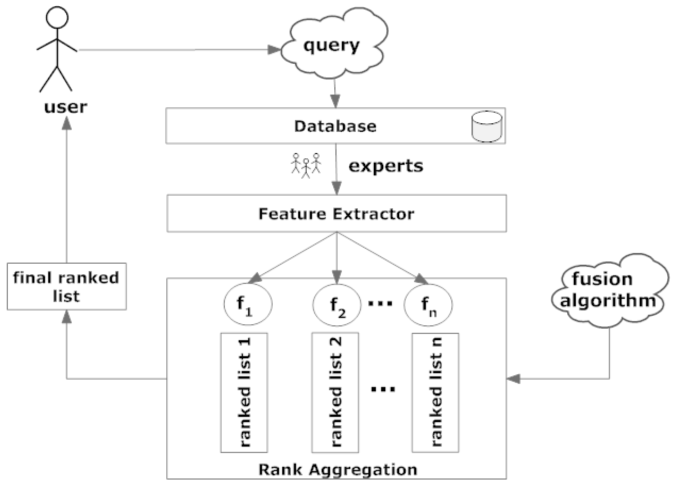

Figure 6.7: Rank Aggregation Framework for Information Retrieval

Given a set of queries $Q = \{q_1, q_2, ..., q_{|Q|}\}$ and a collection of 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 the same ones proposed in Chapter 4 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 candidate experts associated to the query topics, their corresponding ranking lists for each feature and the data fusion technique to be applied. The output produces a ranking score resulting from the aggregation of the multiple feature scores. The relevance of each expert e_j towards the query q is determined through this aggregated score.

6.2 Existing Unsupervised Rank Aggregation Approaches

As already described in Chapter 3, most of the popular works in expert finding are focused on generative probabilistic models which are built upon co-occurrences between the candidates identifiers (such as names and email addresses) and the query topics.

The usage of data fusion techniques has grown increasing interest in the information retrieval community as well as in the expert finding problem (Macdonald and Ounis, 2008).

In what concerns the expert finding problem, Macdonald and Ounis (2008) developed a voting approach for expert finding which used data fusion techniques to rank the candidates. In their framework, the system started just like a normal document based approach, by gathering the top N documents associated with the query topics. Next, each time a candidate appears associated to one of those documents, he receives a vote which expresses his relevance towards the query topic. The final ranking of the candidate was computed through the usage of data fusion techniques. More specifically, the ranking of a candidate was given by the aggregation of the sum of all the computed scores from the documents in which he was present co-occurring with the query topics. Macdonald and Ounis (2008) experimented a set of 12 different data fusion techniques in order to uncover which ones were more suitable for the task of expert finding. They concluded that for the task of expert finding in an organizational environment, the CombMAX data fusion algorithm (i.e., the highest score of a candidate in a set of documents associated to him) achieved very good results.

In what concerns the data fusion techniques proposed in the literature, in the scope of the voting social theory, Riker (1988) suggested a classification to distinguish the different existing data fusion algorithms into two categories, namely the positional and the majoritarian methods.

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. The most representative positional algorithms are the borda count fuse (de Borda, 1781) and the reciprocal rank fuse (Voorhees, 1999).

The majoritarian algorithms are characterized by a series of pairwise comparisons between candidates. That is, the winner is the candidate which beats or ties with all other candidates by comparing their scores between each other. The most representative majoritarian algorithm is probably the Condorcet fuse method proposed by Montague and Aslam (2002). However, there have been other proposed methods based on Markov chain models (Dwork et al., 2001) as well as techniques from multicriteria decision theory (Farah and Vanderpooten, 2007).

On the other hand, Fox and Shaw (1994) 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 well known CombSum and CombMNZ algorithms, which will be detailed in the next section.

6.3 The Unsupervised Rank Aggregation Approach for Expert Finding

Since data fusion can aggregate the rankings of the several individual features for each candidate, has the advantage of not reflecting the tendency of a single feature, but the combination of them, resulting in a more reliable and accurate ranking.

In this work, it was developed a set of experiments with well known data fusion algorithms of the information retrieval literature, such as the CombSUM and the CombMNZ approaches, as well more recent advancements in the state of the art, such as condorcet fusion.

All the data fusion techniques applied in the experiments required normalized sums for the different features. So, to perform the normalization, it was applied the Min-Max Normalization procedure, which is given by Equation 6.30.

$$NormalizedValue = \frac{Value - minValue}{maxValue - minValue} \tag{6.30}$$

6.3.1 The CombSUM, CombMNZ and CombANZ Algorithms

The CombSUM, CombMNZ and CombANZ are rank aggregation algorithms, based on normalized relevance scores, originally proposed by Fox and Shaw (1994).

These algorithms are used to aggregate the information gathered from different sources (i.e., different features) in order to achieve more accurate ranking results than using individual scores. 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 6.31.

$$CombSUM(e,q) = \sum_{j=1}^{k} score_j(e,q) \tag{6.31}$$

Similarly, the CombMNZ score of an expert e for a given query q is defined by Equation 6.32, where r_e is the number of systems which contribute to the retrieval of the candidate.

$$CombMNZ(e,q) = CombSUM(e,q) \times r_e \tag{6.32}$$

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 just like illustrated in Equation 6.33

$$CombANZ(e,q) = \frac{CombSUM(e,q)}{r_e} \tag{6.33}$$

6.3.2 The Borda Fuse

This is a positional rank aggregation method which was originally proposed by de Borda (1781) 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.

Example: Suppose that we have three different features, f_1, f_2 and f_3 and a collection of experts composed by e_1, e_2, e_3, e_4 and e_5. For a given query q, suppose that we have the following ranking lists:
$f_1 = (e_1, e_3, e_2, e_5, e_4)$
$f_2 = (e_3, e_4, e_1, e_2, e_5)$
$f_3 = (e_2, e_1, e_4, e_3, e_5)$

The Borda Fuse (BF) of an expert e_j is computed by summing all the Borda Fuse scores of that expert for each one of the individual features in the following way:

$BF(e_j) = \sum_{i=1}^{3} bf_i(e_j)$

And the computation of the Borda Fuse of each expert is given as follows:

$BF(e_1) = 5 + 3 + 4 = 12$
$BF(e_2) = 3 + 2 + 5 = 10$
$BF(e_3) = 4 + 5 + 2 = 11$
$BF(e_4) = 1 + 4 + 3 = 8$
$BF(e_5) = 2 + 1 + 1 = 4$

And therefore the final ranked list is given by: $e_1 > e_3 > e_2 > e_4 > e_5$

6.3.3 The Reciprocal Rank Fuse

This is a positional rank aggregation method which was originally proposed by Voorhees (1999) 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 feature. The following equation shows the computation of the reciprocal rank scores of some expert e_j using the ranking position of each feature, f_i, from a set of features (i=1...n).

$$RR(e_j) = \sum_{i=1}^{n} \frac{1}{position(f_i(e_j))} \tag{6.34}$$

Example: Considering the previous example, the Reciprocal Rank Fuse (RR) of an expert e_j is computed by summing all the Reciprocal Rank scores of that expert for each one of the individual features, through Equation 6.34, in the following way:

$RR(e_1) = 1 + 1/3 + 1/2 = 1.8333$
$RR(e_2) = 1/3 + 1/4 + 1 = 1.5833$
$RR(e_3) = 1/2 + 1 + 1/4 = 1.7500$
$RR(e_4) = 1/5 + 1/2 + 1/3 = 1.0333$
$RR(e_5) = 1/4 + 1/5 + 1/5 = 0.6500$

And therefore, the final ranked list is given by : $e_1 > e_3 > e_2 > e_4 > e_5$

6.3.4 The Condorcet Fusion

This is a majoritarian rank aggregation method which was originally proposed by Montague and Aslam (2002) 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 provided by Algorithm 5. 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,

Algorithm 5 Condorcet Fusion for Expert Finding

```
Input: pair of experts $e_1$, $e_2$
Output: number of wins and losses of expert $e_1$ towards expert $e_2$
1: $win \leftarrow 0$
2: $lose \leftarrow 0$
3: for each individual feature $f_i$ do
4:     if $f_i$ ranks expert $e_1$ above expert $e_2$ then
5:         $win \leftarrow win + 1$
6:     end if
7:     if $f_i$ ranks expert $e_2$ above expert $e_1$ then
8:         $lose \leftarrow lose + 1$
9:     end if
10: end for
11: return $win, lose$
```

then they are tied (Bozkurt et al., 2007).

6.4 Summary

This chapter argued that rank aggregation methods, based on data fusion techniques, provide significant advantages over the proposed approaches based on languages models from the state of the art. It introduced the general rank aggregation framework for expert search as well as some state of the art approaches, which explore the problem of expert finding as a rank aggregation problem. Finally, it described representative algorithms of the information retrieval literature which were experimented in this work.

The algorithms which were tested in the scope of this work fall into one of three methods, namely the positional methods, the majoritarian methods and the rank aggregation methods.

Positional methods determine the highest ranked candidate by assigning to him a number of votes which corresponds to the position that he occupies in the ranking list.

Majoritarian methods determine the highest ranked candidate through a series of pairwise comparisons between candidates. The winner is the candidate which beats all other candidates by comparing their ranking scores.

The rank aggregation methods determine the highest ranked candidate by simply combining his ranking scores from all the participating systems.

The positional methods which were covered this work were the Borda Fuse and the Reciprocal Rank Fuse algorithms. 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. On the other hand, Reciprocal Rank Fuse determines the highest ranked expert by assigning to each individual candidate a score which corresponds to the inverse of its position in a ranked list given by each feature.

The majoritarian algorithm tested was the Cordorcet Fusion, which 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.

Finally, the score aggregation algorithms tested were the CombSUM, CombMNZ and the CombANZ algorithms. These three algorithms are very similar. CombSUM determines the highest ranked candidate by summing of the normalized scores received by the candidate in each individual ranking. CombMNZ is like CombSUM but the candidate score is also multiplied by the number of systems which contribute to its retrieval. And CombANZ is the same as CombMNZ, but instead of multiplying the number of systems, it divides them.

Chapter 7

Validation and Experimental Results

This chapter validates the main hypothesis behind this work, stating that learning to rank for expert finding is a sound approach to combine multiple estimators of expertise in an optimal way, this way outperforming the current state of the art. It describes the datasets used, as well as all the software packages which enabled the development of this work.

It also describes the set of experiments which were carried out for both proposed approaches for the task of expert finding proposed in this thesis, namely the supervised learning to rank approach and the rank aggregation approach.

This chapter is organized as follows. Section 7.1 describes the datasets used by both experiments as well as important software tools used. Section 7.2 presents the learning to rank approach for expert finding by giving an overview of how the system was trained and tested. It also mentions how the parameters of the various learning to rank algorithms were chosen. Finally, it presents a discussion of the results obtained with the various algorithms tested.

Section 7.3 presents the rank aggregation approach of expert finding by giving an overview of how the system was validated. It also presents the results obtained for the different techniques tested, as well as a brief discussion of those results.

7.1 Validation

In this section, it is described how the approaches proposed in this thesis, namely the supervised learning to rank approach (Chapter 5) and the rank aggregation approach (Chapter 6), were validated. It starts by describing the datasets used for the purposes of the experiments, as well as the tools used to develop the prototype.

7.1.1 DataSets used in the Experiments

The validation of the prototype required a sufficiently large repository of textual contents describing the expertise of individuals within a specific area. In this work, a dataset for evaluating expert search in the Computer Science research domain was used, 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 (Sidiropoulos and Manolopoulos, 2005, 2006) and expert search (Deng et al., 2008, 2011). 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 7.3 provides a statistical characterization of the DBLP dataset.

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 7.3: Statistical characterization of the DBLP dataset used in our experiments

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, it was used the relevant judgements provided by Arnetminer[2] which have already been used in other expert finding experiments by Yang et al. (2009). 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 and from other people's

[1] `http://www.arnetminer.org/citation`
[2] `http://arnetminer.org/lab-datasets/expertfinding/`

relevance judgements. Table 7.4 shows the distribution for the number of experts associated to each topic, as provided by Arnetminer.

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 7.4: Characterization of the Arnetminer dataset of Computer Science experts.

7.1.2 Implementation

For both experiments, it was implemented the methods responsible for computing the features listed in the Chapter 4 of this work, using *Microsoft SQL Server 2008*. This relational database server, together with its full text search capabilities, enabled the computation of the textual similarity features quickly and efficiently. On the other hand, the PageRank features were computed using the Java LAW[1] package.

In order to perform statistical significance tests, it was used a perl script made available by Smucker et al. (2007) which performs a two sided paired randomization test.

In Chapter 4, it was presented a set of features which are based in the author's institutions. The Microsoft Academic Research[2] web site, where it has been put a lot of effort in the construction of a world academic database, was used to collect this information. It was developed a set of $XQuery$ functions which enabled the extraction of such information.

In order to access information and to perform computations, all the gathered data was stored in a database. Figure 7.8 illustrates the entity relation diagram which shows the structure of the database used.

7.2 Learning to Rank for Expert Finding

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

[1]http://law.dsi.unimi.it/software.php
[2]http://academic.research.microsoft.com/

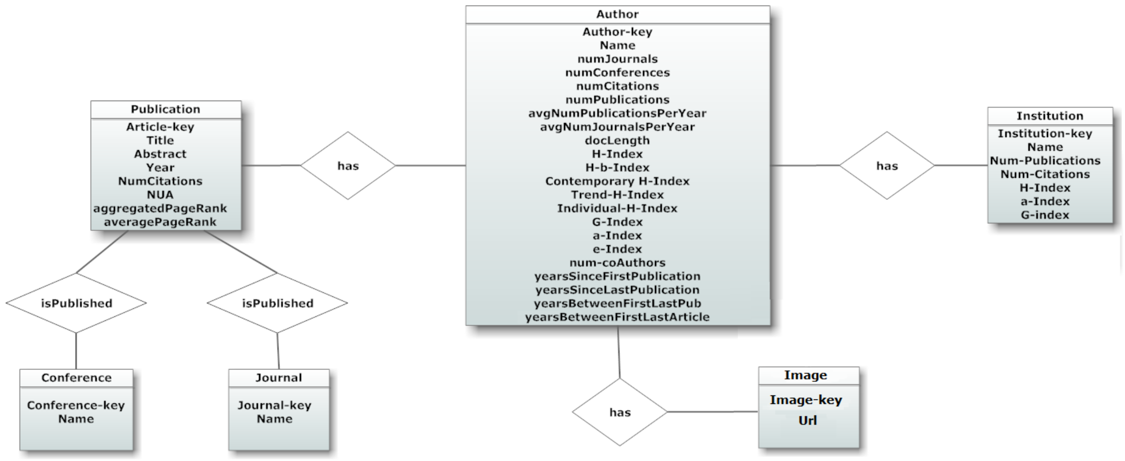

Figure 7.8: Entity relation diagram of the data collected

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. It was used the RankLib[1] software package developed by Van Dang, as well as the SVMrank[2] implementation by Joachims (2006), the SVMmap[3] implementation by Yue et al. (2007) and the Additive Groves[4] algorithm implemented by Sorokina et al. (2007).

The algorithms contained in the RankLib package are AdaRank, RankNet, RankBoost and Coordinate Ascent, already described in Chapter 5 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 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 expert authors 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. (2008), 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, very high features will tend to ignore the values of very low features, this way leading to an unfair learning process.

[1]`http://www.cs.umass.edu/~vdang/ranklib.html`
[2]`http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`
[3]`http://projects.yisongyue.com/svmmap/`
[4]http://additivegroves.net/#downloads

- **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.

In order to train the system, the 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.

7.2.1 Parameter Estimation

Chapter 5 presented a set of learning to rank algorithms which required the manual optimization of some specific parameters (e.g., parameter C in SVMmap and SVMrank). The choice of these parameters is crucial in order to achieve the best performance.

The most naive approach of parameter search is the grid search method. In this approach, it is placed a grid over the parameter space and the data is evaluated at every grid intersection, returning the parameters which lead to maximum performance of the learning algorithm (Metzler and Croft, 2007). However, grid search has the problem of being unbounded, since an infinite set of parameters are available to be tested. To overcome this issue, the parameter search was restricted using the boundaries suggested by Hsu et al. (2010) for the SVM based algorithms ($C \in \{2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}\}$ and $\gamma \in \{2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}\}$). For the other approaches (AdaRank, Coordinate Ascent, RankBoost and RankNet), the grid search was stopped when the results obtained, in terms of MAP, started to converge.

The grid search approach is a very naive method for finding parameters in the parameter space. In fact, there are several advanced parameter search algorithms in the literature which do not have a heavy computational cost. The motivation behind the usage of the grid search approach in this work is that the computational time required by any of the other advanced methods is almost the same as using grid search, since in our experiments, at most two parameters were searched. In addition, many of the advanced search parameter methods in the literature perform approximations which can be avoided by the direct parameter search of the grid search approach.

Table 7.5 shows the obtained results when applying grid search in the learning to rank algorithms tested.

Parameters	AdaRank	Coordinate Ascent	RankBoost	RankNet
# Iterations	400	100	300	30
# Threshold Candidates	-	-	40	-
# Random Restarts	-	5	-	-
# Nodes in Hidden Layer	-	-	-	50

Table 7.5: Parameters found with the grid search approach

Since the SVMmap and SVMrank algorithms are based in the formalisms of the Support Vector Machines, we still had to take into consideration the type of kernel to use.

As described in Appendix A, Radial Basis Function (RBF) kernels are the most widely non-linear kernel used. The RBF kernel contains the C parameter from the linear kernel and a γ parameter which determines the area of influence that the center support vector has over the data space.

In order to determine which kernel performed better (the linear kernel or the RBF kernel), we decided to compare them by making some experiments with the learning to rank algorithms based on the formalisms of the Support Vector Machines, namely SVMrank and SVMmap.

In what concerns the linear kernel, Figure 7.9 shows the SVMmap algorithm trying to optimize the mean average precision score in each iteration, while SVMrank has more stable scores. The SVMmap with a linear kernel achieved an optimum score for $C = 600$ and SVMrank for $C = 900$.

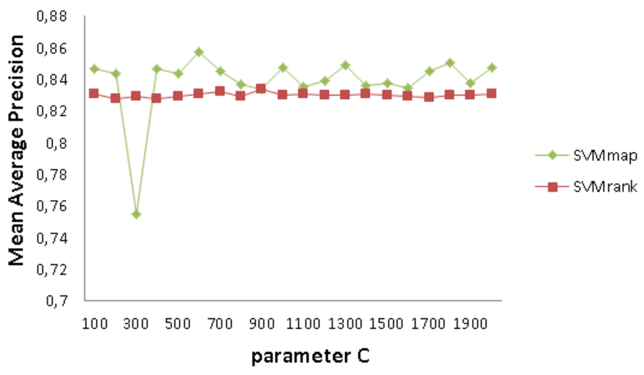

Figure 7.9: Linear kernel grid-search. For SVMmap, C achieved the value 600 (MAP = 0.8573) while, for SVMrank, C achieved a value of 900 (MAP = 0.8339)

For the non-linear kernel, we only made experiments with the SVMmap algorithm, because SVM-rank did not converge in convenient time, just like predicted by Joachims (2006). The best score was found for $C = 8$ and $\gamma = 0.125$, with a MAP score of 0.8702, just like illustrated in Figure 7.10.

Since the non-linear SVMmap approach achieved better results than the linear one, the rest of this work will consider the non-linear kernel results whenever SVMmap is mentioned.

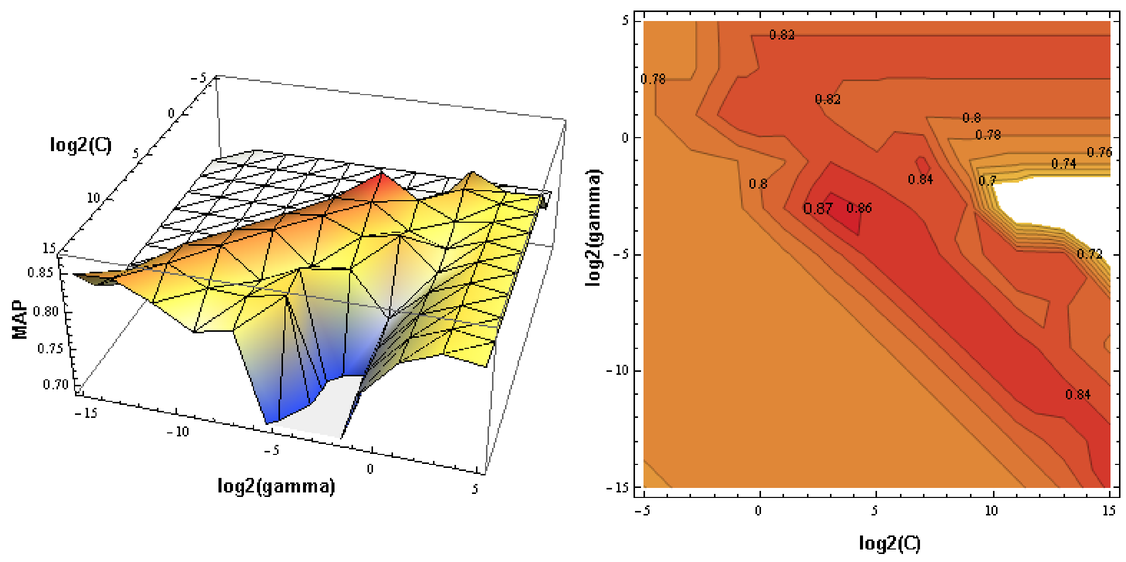

Figure 7.10: Function representing the grid search performed with SVMmap (left). Contours of the same function (right). The best values found were $C = 8$ and $\gamma = 0.125$

7.2.2 Results and Discussion

In this section, it is described the results obtained in the supervised learning to rank experiments for the enriched version of the DBLP dataset. We made experiments with representative algorithms of the pointwise, pairwise and listwise approaches from the learning to rank literature. More specifically, the listwise methods tested were the Coordinate Ascent, AdaRank and SVMmap. The pairwise approaches were RankNet, RankBoost and SVMrank. And Finally, the pointwise approach tested was the Additive Groves algorithm.

	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 7.6: Results the various learning to rank algorithms tested

Table 7.6 presents the obtained results over the DBLP dataset. These results show that the pointwise approach Additive Groves outperforms all other pairwise and listwise learning to rank algorithms, in terms of MAP, and therefore provides 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 outperforms Additive Groves when ranking the top 5 experts (P@5), showing that this listwise method can also provide a successful approach to be used in the context of expert finding.

Table 7.6 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 in academic digital libraries.

As for the worst results obtained, Table 7.6 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 7.11 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 thesis, it was 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, namely AdaRank, Coordinate Ascent, RankNet, RankBoost, SVMrank and SVMmap. 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, Additive Groves 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, it was 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 7.7 shows the obtained results, also presenting the previous results reported by Yang et al. (2009), as well as the representative state of the art approaches proposed by Balog et al. (2006), over the same dataset.

	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 (Balog et al., 2006)	-	0.0250	-	0.0180	0.0060	-
Balrog's Model 2 (Balog et al., 2006)	-	0.5750	-	0.4600	0.3915	-
Expert Finding (Yang et al., 2009)	0.5500	0.6000	0.6333	-	0.6356	-

Table 7.7: The results obtained with the Additive Groves algorithm for the different sets of features and comparison with other approaches.

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 proposed by Yang et al. (2009).

Table 7.7 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. (2006), namely the candidate-based Model 1 and the document-based Model 2, already described in Chapter 3 of this work. These results were reported by Deng et al. (2011) which have modified the original code developed by Balog et al. (2006) 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 thesis. 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 learning to rank algorithms proposed in this thesis 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, it was applied a two sided 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, Additive Groves using the full set of features was not more significant than the Additive Groves algorithm using individually (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 7.11 plots the obtained average precision scores in each of the individual query topics for the best performing approach, namely Additive Groves algorithm with the combination of all features. The figure presents the query topics in the same order as they are given in Table 7.4. 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.

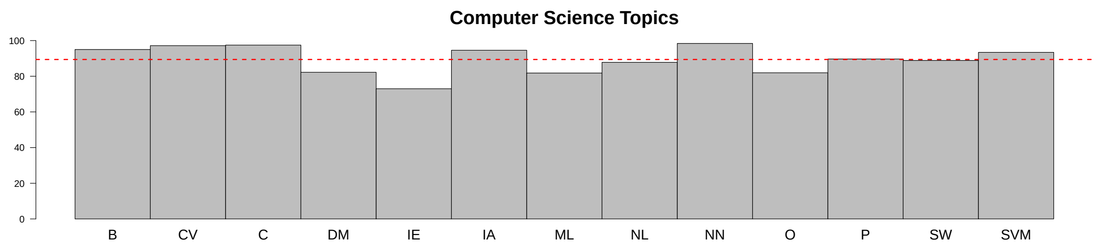

Figure 7.11: Average precision over the different query topics using the Additive Groves algorithm.

Finally, Table 7.8 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 his publications' titles and abstracts contributing for a misclassification.

	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 7.8: Top five people returned by the system for four different queries using the Additive Groves algorithm.

7.3 Rank Aggregation for Expert Finding

Since relevant judgements for the expert finding task in digital libraries are very hard to find, we decided to perform rank aggregation experiments 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, also outperforming the representative approaches of the state of the art.

To validate this hypothesis, a prototype expert search system was built by implementing six different data fusion algorithms based in positional methods, majoritarian methods and score aggregation techniques. 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 already been described in Chapter 6 of this work.

To validate the different rank aggregation algorithms, since the Arnetminer dataset 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.

7.3.1 Results and Discussion

This section describes the results obtained in the rank aggregation experiments for the enriched version of the DBLP dataset. Experiments were performed with representative score aggregation algorithms, positional methods and majoritarian approaches from the information retrieval literature. More specifically, the score aggregation algorithms tested were the CombSUM, CombMNZ and CombANZ algorithms. The positional methods were the Borda Fuse and the Reciprocal Rank Fuse. And finally, the majoritarian approach tested was the Condorcet Fusion algorithm.

	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 7.9: Results of the various data fusion algorithms tested

Table 7.9 presents the obtained results over the DBLP dataset. The obtained results show that the CombMNZ rank aggregation technique outperforms all other algorithms, in terms of MAP, showing that rank aggregation methods provide 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 7.6 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 the Borda Fuse algorithm 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 7.6 shows that the 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 7.11 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.

It was also applied a two sided paired randomization test in order to see if the best performing algorithm, more specifically Cordocet Fusion, was indeed more significant than all other algorithms, namely CombSum, CombMNZ, CombANZ, Borda Fuse and Reciprocal Rank Fuse. 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, the Condorcet Fusion algorithm was not more significant than all the other remaining 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, it was attempted to measure the impact of the different types of ranking features on the quality of the results. Using the best performing data fusion technique, namely the Condorcet Fusion algorithm, it was 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 7.10 shows the obtained results, also presenting the previous results reported by Yang et al. (2009) and Deng et al. (2011), as well as the representative state of the art approaches proposed by Balog et al. (2006), 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

	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 (Balog et al., 2006)	-	0.0250	-	0.0180	0.0060	-
Balrog's Model 2 (Balog et al., 2006)	-	0.5750	-	0.4600	0.3915	-
Query-Sensitive AuthorRank (Deng et al., 2011)	-	**0.6800**	-	0.5350	**0.4906**	-
Expert Finding (Yang et al., 2009)	0.5500	0.6000	0.6333	-	0.6356	-

Table 7.10: The results obtained with the Condorcet algorithm for different sets of features and comparison with other approaches.

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 outperform the representative state of the art approaches proposed by Balog et al. (2006), namely the candidate-based Model 1 and the document-based Model 2. On the other hand, the rank aggregation approach proposed in this thesis has a slightly poor performance when compared to the query sensitive AuthorRank model proposed by Deng et al. (2011). One reason for these results is that Deng et al. (2011) 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 7.10 also shows that this rank aggregation framework did not outperform the previously learning to rank approach proposed by Yang et al. (2009). This was already expected, since supervised approaches find optimal and more accurate ranking models than 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 thesis 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, it was 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 Condorcet Fusion algorithm using 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).

Figure 7.12 plots the obtained average precision in each of the individual query topics for the best performing approach, namely the Condorcet Fusion algorithm with the combination of all features. The figure presents the query topics in the same order as they are given in Table 7.4. Again, the horizontal dashed line corresponds to the MAP obtained in the same experiment. The results show that there are some significant variations in performance for the different queries.

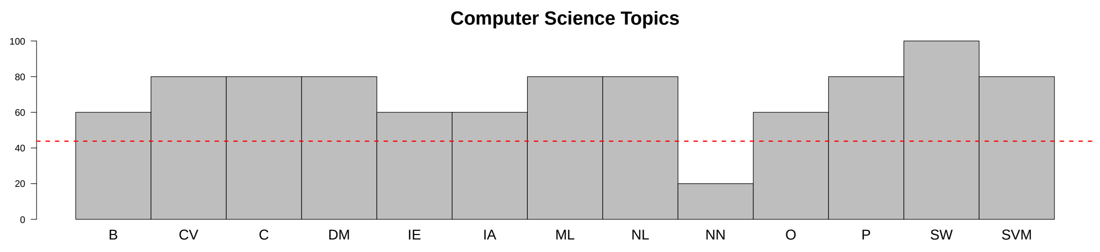

Figure 7.12: Average precision over the different query topics using the Condorcet Fusion algorithm.

Finally, Table 7.11 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 Data Mining, Computer Vision and Semantic Web. However, worse results were returned for the query Neural Networks. After analysing the top five authors returned for this query, it was found that many of these researchers have lots of publications in this field. For instance, the top author returned, Jun Wang, has 129 articles and 316 publications in conferences, where 164 of his publications contain the query topics. And the same happens for the other authors returned by the system. The task of expert finding is very challenging, since it is very difficult to provide relevance judgements stating who in fact is an expert in some topic, because opinions may vary between people. The Arnetminer dataset was built gathering people from important program committees and from other researchers judgements. So it is normal that some important people may not be contained in these relevance judgements. In addition, since the author Jun Wang was not present in the Arnetminer's dataset, it was considered not relevant and therefore the system made a misclassification, although, according to his curriculum, he has high knowledge in the topic.

Best Results			Worst Results
Data Mining	**Computer Vision**	**Semantic Web**	**Neural Networks**
Christos Faloutsos	Ramesh Jain	Elisa Bertino	Jun Wang
Philip S. Yu	Takeo Kanade	Dieter Fensel	C. Lee Giles
Jiawei Han	Thomas S. Huang	Amit P. Sheth	Zhi-Hua Zhou
Hans-Peter Kriegel	Shree K. Nayar	Rudi Studer	Witold Pedrycz
Raghu Ramakrishnan	Alex Pentland	James A. Hendler	Jude W. Shavlik

Table 7.11: Top five people returned by the system for four different queries using the Condorcet Fusion algorithm.

7.4 Summary

This chapter presented a discussion of the results obtained in the experiments using learning to rank and rank aggregation approaches.

Both experiments used the DBLP computer science dataset as sources of evidence of the knowledge of the authors, and the Arnetminer dataset for validation purposes.

For implementation, several Java software packages were used as well as the Microsoft SQL Server database engine to compute the features described in Chapter 4. All the information contained in the dataset was stored in a database as well as external information regarding the author's affiliations. This information was extracted from Microsoft Academic Research website.

The results obtained in the supervised learning to rank approach showed that the pointwise approach Additive Groves outperformed all other learning to rank approaches. However, the SVMmap listwise method also achieved very competitive results. These results also showed that the listwise approach AdaRank and the pairwise approach RankNet performed poorly. AdaRank greedy feature selection method and RankNet gradient descent, which can get stuck in a local minimum, made these two approaches not useful for the task of expert finding in digital libraries.

When testing the combination of the different sets of features with the best performing supervised algorithm, namely Additive Groves, the results showed that the combination of all sets of features achieved the best results. On the other hand, the textual features combined with graph features or combined with profile features achieved very competitive results as well, showing that the textual features are very important for the model computed by the Additive Groves algorithm.

In what concerns the rank aggregation experiment, the results showed that the Condorcet Fusion majoritarian approach, outperformed all others, although the score aggregation approach

CombMNZ also achieved very competitive results. CombMNZ and Condorcet Fusion are two state of the art algorithms and these results also supported it. These results also showed that the CombANZ method performed very poorly, because in its definition it affects negatively experts which have lots of systems contributing for their retrieval. Another finding is that both positional methods tested turned out to have the same results. This is not very surprising since the Borda Fuse method and the Reciprocal Rank Fuse method are nearly the same. One uses directly the position of the documents and the other one uses the reciprocal rank of the position. It turns out that in this experiment the final ranked lists were the same.

When testing the combination of the different sets of features with the best performing data fusion technique, namely the Condorcet Fusion algorithm, the results showed that the combination of all sets of features achieved the best results. On the other hand, the features based on co-authorship and citation graphs also achieved very competitive results. This was an expected result, since the DBLP dataset has a very rich citation link information.

Although this rank aggregation framework did not outperformed the previously learning to rank approach proposed by Yang et al. (2009), 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, and therefore this rank aggregation algorithm also provides very competitive results for the expert finding literature.

Chapter 8

Conclusions and Future Work

This is the ending chapter of this work. The top contribution presented in this thesis is the learning to rank framework for expert finding in digital libraries. An approach which so far is almost nonexistent in the expert finding literature. In this thesis, it was made a careful study of the existing learning to rank algorithms in order to determine which ones are more suitable for the expert finding task. My hypothesis, which stated that learning to rank approaches can be used as a sound a approach for combining various estimators of expertise in an optimal way, was validated. This work outperformed the existing known supervised state of the art approach, proposed by Yang et al. (2009), achieving a mean average precision result of 89.40%. This approach also outperforms the representative state of the art approaches proposed by Balog et al. (2006). All the goals that I pretended to achieve with this thesis were overcome.

In this chapter it is presented the final remarks of this work. Section 8.1 states the main contributions of this work, by summarizing the algorithms tested in each one of the experiments performed and summarizing the different sets of features which were proposed in this thesis. Section 8.2 presents the conclusions of this thesis by making an overview of this work. Finally, Section 8.3 presents some directions of future work that this thesis can point to.

8.1 Contributions for Expert Finding

The main contributions of this thesis are summarized in the following subsections.

8.1.1 A Set of Features to Estimate Expertise

In this work, it was presented a set of features which are intended to characterize the candidate's knowledge. The definition of these features has significant importance, because the learning machines tested in this work, as well as the rank aggregation approaches, required a set of features which were able to identify whether or not a candidate is relevant given a specific query topic. If these features are not able to represent the knowledge of a candidate, then the whole learning to rank approach collapses. In the scope of this thesis, three different sets of features were defined, based on different sources of evidence. It was proposed features in a textual similarity level, where it was considered traditional information retrieval techniques which measure the co-occurrences between the query topics and the documents associated to a candidate. It was also defined profile information features which measure the total publication record of a candidate throughout his career, under the assumption that highly prolific candidates are more likely to be considered experts. Finally, it was defined a set of graph features which enable the construction of link-based structures which admit the computation of the scientific impact of a candidate in the scientific community. In this set of features, it was also defined a set of academic indexes based in the scientometrics community, in order to determine such scientific impact. The effectiveness of these features have already been published as part of some research works by Moreira, Martins and Calado (2011) and Moreira, Calado and Martins (2011).

8.1.2 A Supervised Learning to Rank Approach for Expert Finding

The top contribution of this thesis is given by the supervised learning to rank prototype.

The learning algorithms experimented covered the three types of approaches of the learning to rank framework, namely the pointwise, pairwise and listwise approaches.

The pointwise approach experimented was the Additive Groves method. This algorithm builds a single small tree, i.e. a grove, from the training data. 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 pairwise approaches tested were the RankBoost, RankNet and SVMrank algorithms.

RankBoost trains one weak ranker at each round and the final model corresponds to a linear combination of these rankers. In each iteration, RankBoost decreases the weights of the pairs of experts which were well classified and increases the weights of the misclassified ones, in order to give more attention at these misclassifications in the following round.

RankNet uses a multilayer neural network with a modified 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 number of misclassified expert pairs by minimizing the value of the cost entropy function by adjusting each weight in the network with the backpropagation algorithm.

SVMrank builds a ranking model, through the formalism of support vector machines, by attempting 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.

Finally, the listwise algorithms experimented in this work were AdaRank, Coordinate Ascent and SVMmap. AdaRank works just like RankBoost, the difference resides in how the weak rankers are chosen. At each iteration, AdaRank increases the weight of the experts which performed poorly concerning some information retrieval evaluation metric and decreases the weights of the instances which performed very well. This way, the next iteration will only be focused in the misclassified experts.

Coordinate Ascent is an algorithm which attempts to solve unconstrained optimization problems. The algorithm iteratively holds all features except for one, which is fixed, and tries to reoptimize the problem with respect to the fixed feature.

SVMmap builds a ranking model under the formalisms of support vector machines and tries to minimize a loss function which measures the difference between the performance of a perfect ranking and the minimum performance of an incorrect ranking.

Part of this experiment has already been published by Moreira, Calado and Martins (2011).

8.1.3 A Rank Aggregation Approach for Expert Finding

In this work, it was presented the rank aggregation prototype based in data fusion techniques. The data fusion techniques experimented covered the three types of approaches of the rank aggregation framework, namely the positional methods, majoritarian methods and score aggregation methods.

The positional methods that covered this work were the Borda Fuse and the Reciprocal Rank

Fuse algorithms.

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.

The Reciprocal Rank Fuse algorithm 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.

The majoritarian algorithm tested was the Cordorcet Fusion, which 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.

Finally, the score aggregation algorithms tested were the CombSUM, CombMNZ and the CombANZ algorithms.

These three algorithms are very similar. CombSUM determines the highest ranked candidate by summing the normalized scores received by the candidate in each individual ranking. CombMNZ is like CombSUM, but the candidate score is also multiplied by the number of systems which contribute to its retrieval. And CombANZ is the same as CombMNZ, but instead of multiplying the number of systems, it divides them.

Part of this experiment has already been published by Moreira, Martins and Calado (2011).

8.2 Conclusions

The effectiveness of the Learning to Rank approach for expert finding.

The results presented in Chapter 7 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. (2009) which, as far as I 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.

It was 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 in the final ranked list 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[1]. 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.

The experiments also showed that AdaRank and RankNet algorithms performed poorly when compared to the other approaches tested and therefore are not suitable for the task of expert finding in digital libraries. These results were also surprising, since the AdaRank algorithm is a listwise method which aims at optimizing an information retrieval metric. However, its greedy approach for feature selection turns out to be not very effective in this dataset. RankNet, on the other hand, suffers from the disadvantages of the gradient descent method, that is, it can be stuck in a local minimum and therefore the optimization process is compromised.

The effectiveness of the Rank Aggregation approaches for expert finding.

The results presented in Chapter 7 showed that rank aggregation approaches also provide reasonable results for the task of expert finding in digital libraries, since they outperformed some representative state of the art works. 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. The Condorcet Fusion algorithm performed very well due to its pairwise voting procedure which favours all candidates that beat others in a pairwise comparison.

On the other hand, aggregating the ranking scores of each candidate taking into account the number of systems which contributed to its retrieval, also provides a very simple and competitive approach for this task.

Effectiveness of the Features Chosen

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 thesis.

In Chapter 7, it was presented the results obtained with individual features as well as with different

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

combinations of sets of features. One concluded that all sets of features (textual, profile and graph) were required in order to achieve the best results in both experiments. Although that individually, the graph features, as well as the textual features, also achieved very competitive results for the rank aggregation and the supervised learning to rank approaches, respectively.

8.3 Directions of Future Work

In this section, it is presented some directions of future work that this thesis can point to. The directions proposed are based in feature selection methods, in order to improve the current prototype, as well as modifications of the proposed prototype to be able to handle the expert finding problem in an organizational environment. Finally, it points other directions which are not related with the expert finding task, but are very related to the learning to rank formalisms.

8.3.1 Feature Selection

The experiments performed in this thesis required an amount of 60 features to estimate the degree of expertise of an expert. Since we are dealing with big databases, the computation of these features turns the system heavy and slow in run time. It would be very interesting to test feature selection methods which allowed the computation of less features and kept the same performance. Various feature selection methods have been proposed in the literature, but they manly consist in exhaustive experiments where they make combinations of single features in order to achieve better results. This work proposes a set of 60 features and that method is not suitable due to the exponential number of combinations that would be required.

Cohen et al. (2002) proposed a feature selection method for face recognition using a modified version of the Principal Component Analysis, which basically performs an orthogonal projection of the features converting them into values of uncorrelated features. These features are the ones which have bigger impact and are usually much less than the original number of features This approach would be very interesting to experiment in the scope of this work, due to its simplicity and great performance.

8.3.2 Expert Search in Enterprises

The study performed in this work was for the expert finding task for digital libraries. It would be very interesting to apply the algorithms tested in this work in the TREC enterprise track dataset.

In the learning to rank approach for expert finding proposed by Macdonald and Ounis (2011), they achieved the best results using the AdaRank listwise algorithm, turning their approach one of the top contributions for the expert finding task in an organizational environment. However, the experiments performed in this thesis showed that AdaRank was the approach which achieved the worse results. I am very curious to know how the Additive Groves algorithm would perform in such task.

8.3.3 Tasks Beyond Expert Search

The learning to rank study performed in this work can be very useful for other tasks of information retrieval. For instance, in the scope of search engines, Martins and Calado (2010) have proposed a geographic information retrieval model based on SVMmap. It would be interesting to extend their approach to improve retrieval in search engines by combining a set of textual features with a set of geographic features also exploring the learning to rank algorithms tested in this thesis.

Another interesting approach would be in the field of Q&A. In the past, Moreira, Mendes, Coheur and Martins (2011) have developed a natural language understanding module for the conversational agent Edgar[1] using support vector machines to learn a model which is able to classify the questions in order to retrieve the right answers. It would be interesting to extend Edgar's capabilities in order to incorporate learning to rank mechanisms which allowed him to choose the best answer, from a set of possible answers, for a given question.

[1] http://edgar.l2f.inesc-id.pt/index.php

Appendix A

Support Vector Machines for Information Retrieval

Information retrieval essentially deals with ranking problems. An adhoc retrieval system checks the documents of a collection and ranks them according to their degree of relevance to a user query expressing the information need. The major difficulty faced by an IR system resides in the computation of these relevance estimates. The aim of machine learning is to use algorithms which are able to automate information retrieval processes such as classification tasks, reducing inconsistencies which could be introduced by human errors and therefore providing more accurate ranking models.

In order to truly understand the algorithms experimented in this thesis, a background in learning machines is required, more specifically in the domain of Support Vector Machines.

In this appendix, it is presented an overview of this learning framework, detailing its architecture and describing how the learning process is performed.

A.1 Support Vector Machines

Support Vector Machines (SVMs) were originally proposed in the works of Boser et al. (1992) and Cortes and Vapnik (1995). They can be defined as learning machines which construct a N-dimensional decision boundary surface (also called a hyperplane) that optimally separates data into positive examples and negative examples, by maximizing the margin of separation between these examples.

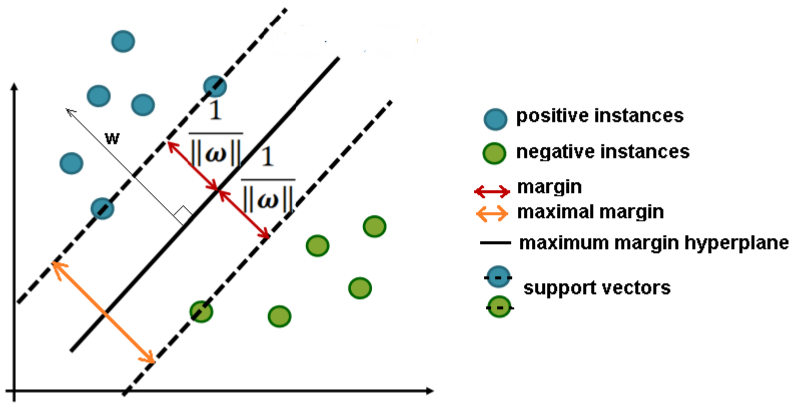

Figure A.13: General SVM classification problem

The Support Vector Machines can be constructed in two different formalisms depending whether the training data can be considered linear separable or non-linear separable.

A.1.1 SVMs for Linearly Separable Data

Following Haykin (2008), given the training sample $\{(x_i, d_i)\}_{i=1}^{N}$, x_i is the feature vector (input) associated to document i and d_i is the desired response (output). The output values are $d_i = +1$ if document i is a positive example and $d_i = -1$ a negative one. Assuming that the training sample can be linear separable, that is, we can optimally separate the training data using a N-dimensional hyperplane, then Equation A.35 represents the hyperplane which can perform such separation.

$$w^T x + b = 0 \tag{A.35}$$

In the above equation, x is an input vector, w the adjustable weight vector normal to the hyperplane and b is a bias which measures how far the hyperplane is to the origin. The *margin of separation* of a SVM is defined as the separation between the hyperplane from Equation A.35. The goal of the SVMs is to construct this hyperplane in a way that its distance to the margin is maximized. The larger this distance, the better the model will generalize and be able to classify unseen data. In other words, we want to choose the largest distance. Figure A.13 shows the general SVM problem. After computing the optimal hyperplane, the classification is performed through Equation A.36.

$$h(x) = sign(w^T x + b) \tag{A.36}$$

Every input vector falling above this hyperplane is classified as a positive instance and an input vector falling bellow this hyperplane is classified as a negative instance.

$$w^T x + b \geq 1 \text{ for } d_i = +1 \tag{A.37}$$

$$w^T x + b \leq 1 \text{ for } d_i = -1$$

In the special case where $w^T x + b = \pm 1$, this generates a support vector that constraints the width of the margin of the classifier. Figure A.13 shows that the maximal margin is given by twice the distance between two parallel support vectors, that is, $\frac{2}{\|w\|}$. It follows that in order to find the maximum margin between them, it is required the SVM to solve the quadratic optimization problem given by Equation A.38.

$$min_{w,b} \frac{1}{2} \|w\|^2 \tag{A.38}$$

$$\text{s.t. } \forall_i \in (i = 1, ..., n) \; y_i(wx_i - b \geq 1)$$

Since the above optimization problem only depends in a subset of the training data, more specifically, we are only interested in the points which constraint the width of the margin, that is the support vectors. Then, the classification function given by Equation A.36 can be rewritten as follows.

$$h(x) = sign(\sum_{i=1}^{N} \alpha_i d_i x_i x - b) \tag{A.39}$$

In the above equation, α_i represent the Lagrange multipliers which were computed in the optimization problem and N is the number of support vectors.

So far, the classification model presented is more suitable for perfectly separable data and therefore it is called the hard margin SVM classifier. Since most of the real world data are not linear separable, Cortes and Vapnik (1995) suggested a modification to the above classifier and proposed the *soft margin* SVM classifier. The authors proposed a modified maximum margin which enabled the existence of mislabeled examples. Figure A.14 shows an example of the classifier.

The basic idea behind this approach is the following. If there is no hyperplane which can perfectly split the positive examples from the negative ones, then the soft margin method will chose a hyperplane that can split the examples as cleanly as possible, allowing some misclassified examples, but maintaining the maximum distance to the nearest cleanly split example.

In order to allow misclassifications, the method uses slack variable, ξ_i, which can measure the the degree of misclassification of the datum x_i. Therefore, the optimization problem of a soft

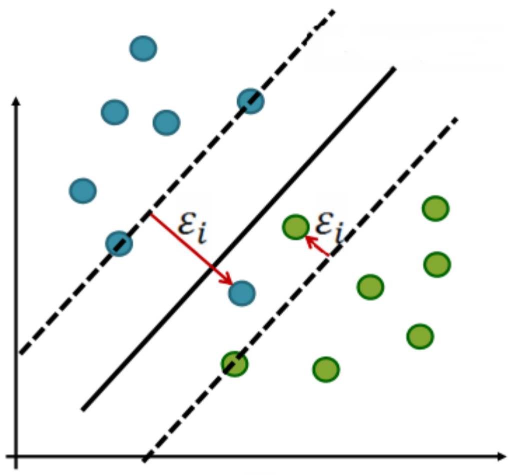

Figure A.14: Soft margin linear SVM classifier

margin classifier can be defined by Equation A.40.

$$min_{w,\xi,b}\frac{1}{2}\left\|w\right\|^2 + C\sum_{i=1}^{N}\xi_i \tag{A.40}$$

s.t. $d_i(w.x_i - b) \geq 1 - \xi, \xi \geq 0$ for any $i(i = 1, ..., n)$

In the above equation, C is a regularization parameter. It 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.

A.1.2 SVMs for Non-Linearly Separable Data

So far we presented a method to compute the optimal hyperplane through a linear classifier. However, the training samples are considered non-linear separable, that is, it is not possible to construct a separating hyperplane without encountering classification errors. In order to solve this problem, Boser et al. (1992) proposed a non-linear classifier by applying a method called the *kernel trick* to maximize the margin of the hyperplane, Figure A.15.

The kernel trick maps the training samples $\{(x_i, d_i)\}_{i=1}^{N}$ into a higher dimensional space called the feature space in the hope that the samples can be linearly separable in the high dimension. The trick in this method consists in avoiding the explicit mapping by only computing dot products

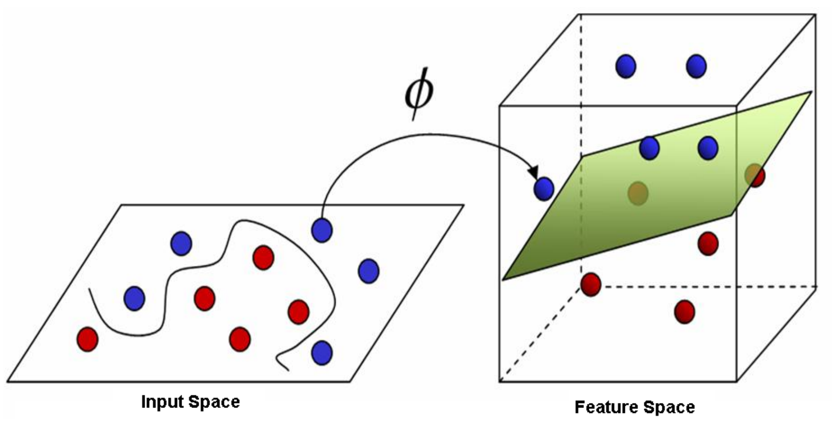

Figure A.15: The non-linear SVM classifier with the kernel trick

between the vectors in the feature space. The mapping must be chosen such that these high dimensional dot products can be computed within the original space by means of a kernel function. Although the classifier in the feature space computes the optimum hyperplane, the transformation back to the input space may be non linear.

Accoording to Haykin (2008), the most common kernel functions used are the following:

- linear kernel
- polynomial kernel
- radial basis function kernel
- two layer perceptron kernel

In this thesis, we focused on the most widely used non-linear SVM kernel, namely the radial basis function kernel.

A.1.3 The Radial Basis Function Kernel

The Radial Basis Function (RBF) is any real valued function which depends on the distance from the origin or some specific center to a data point. The output of the kernel is dependent on the Euclidean distance of x_i from x_j (the first will be the support vector and the second will be the testing data point).

In the radial basis function kernel type of a Support Vector Machine, the number of radial basis functions and their centres are determined automatically by the number of support vectors and

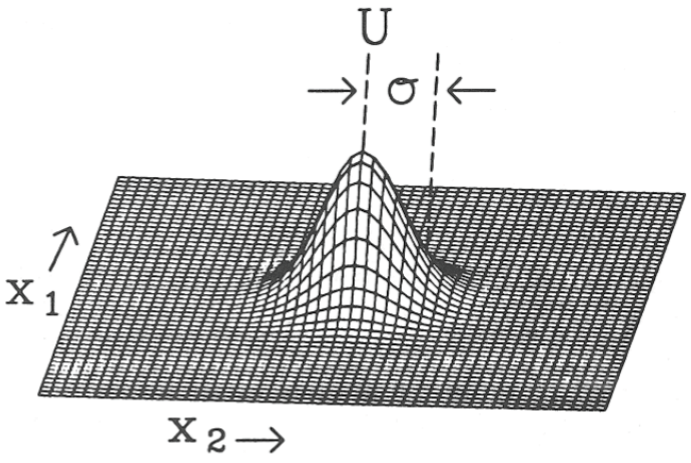

Figure A.16: The radial basis function kernel

their values respectively, in order to minimize an upper bound on the expected test error (Haykin, 2008).

$$k(x_i, x_j) = exp(-\gamma \left\| x_i - x_j \right\|^2), \quad \texttt{for} \quad \gamma > 0 \tag{A.41}$$

In Equation A.41 the parameter γ has to be manually chosen. A larger value of γ will give a smoother decision surface and a more regular decision boundary. This is because an RBF with large γ will allow a support vector to have a strong influence over a larger area. In addition, the support vector will be the centre of the RBF and γ will determine the area of influence this support vector has over the data space. Figure A.16 shows a general Gaussian RBF function.

Bibliography

Baeza-Yates, R. and Ribeiro-Neto, B. (1999), *Modern Information Retrieval*, Addison Wesley.

Balog, K. (2008), People Search in the Enterprise, PhD thesis, University of Amsterdam, Faculty of Science, Mathematics and Computer Science.

Balog, K., Azzopardi, L. and de Rijke, M. (2006), 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)'.

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

Balog, K., Bogers, T., Azzopardi, L., de Rijke, M. and van den Bosch, A. (2007), Broad expertise retrieval in sparse data environments, *in* 'Proceedings of the 30th annual international ACM Conference on Research and Development in Information Retrieval (SIGIR '07)'.

Balog, K. and de Rijke, M. (2008*a*), Combining candidate and document models for expert search, *in* 'Proceedings of the 2008 Text REtrieval Conference (TREC 2008)'.

Balog, K. and de Rijke, M. (2008*b*), Non-local evidence for expert finding, *in* 'Proceeding of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)'.

Banks, M. G. (2006), 'An extension of the hirsch index: Indexing scientific topics and compounds', *Scientometrics* **69**, 161–168.

Bao, S., Duan, H., Zhou, Q., Xiong, M., Cao, Y. and Yu, Y. (2007), Research on expert search at enterprise track of trec 2006, *in* 'Proceedings of the 15th Rext REtrieval Conference (TREC 2006)'.

Batista, P. D., Campiteli, M. G. and Kinouchi, O. (2006), 'Is it possible to compare researchers with different scientific interests?', *Scientometrics* **68**, 179–189.

Belew, R. (2000), *Finding Out About*, Cambridge University Press.

Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992), A training algorithm for optimal margin classifiers, *in* 'Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (COLT '92)'.

Bozkurt, I. N., Gurkok, H. and Ayaz, E. S. (2007), 'Data fusion and bias'.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. and Hullender, G. (2005), Learning to rank using gradient descent, *in* 'Proceedings of the 22nd International Conference on Machine Learning (ICML '05)'.

Campbell, C. S., Maglio, P. P., Cozzi, A. and Dom, B. (2003), Expertise identification using email communications, *in* 'Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM '03)'.

Cao, Y., Liu, J., Bao, S. and Li, H. (2006), Research on expert search at enterprise track of trec 2005, *in* 'Proceedings of the 14th Text REtrieval Conference (TREC 2005)'.

Cao, Z., Liu, T. Q. T.-Y., Tsai, M.-F. and Li, H. (2007), Learning to rank: from pairwise approach to listwise approach, *in* 'Proceedings of the 24th International Conference on Machine learning (ICML '07)'.

Chakrabarti, S., Khanna, R., Sawant, U. and Bhattacharyya, C. (2008), Structured learning for non-smooth ranking losses, *in* 'Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD '08)'.

Chen, H., Shen, H., Xiong, J., Tan, S. and Cheng, X. (2006), Social network structure behind the mailing lists: Ict-iiis at trec 2006 expert finding track, *in* 'Proceedings of the 15th Text REtrieval Conference (TREC 2006)'.

Chen, P.-J., Xie, H., Maslov, S. and Redner, S. (2007), 'Finding scientific gems with google's pagerank algorithms', *Journal of Informetrics* **1(1)**, 8–15.

Cohen, I., Tian, Q., Zhou, X. S. and Huang, T. S. (2002), Feature selection using principal component analysis, *in* 'Proceedings of the 2002 International Conference on Image Processing (ICIP '02)'.

Cortes, C. and Vapnik, V. (1995), 'Support-vector networks', *Machine Learning* **20**, 273–297.

Cossock, D. and Zhang, T. (2006), Subset ranking using regression, *in* 'Proceedings of the 19th International Conference on Learning Theory (COLT '06)'.

Crammer, K. and Singer, Y. (2001), Pranking with ranking, *in* 'Advances in Neural Information Processing Systems (NIPS '01)'.

Craswell, N., de Vries, A. P. and Soboroff, I. (2005), Overview of the trec-2005 enterprise track, *in* 'Proceedings of the 14th Text REtrieval Conference (TREC 2005)'.

Craswell, N., Hawking, D., Vercoustre, A.-M. and Wilkins, P. (2001), P@noptic expert: Searching for experts not just for documents, *in* 'Ausweb'.

de Borda, J.-C. (1781), *Mémoire sur les Élections au Scrutin.*, Histoire de l'Académie Royale des Sciences.

Deng, H., King, I. and Lyu, M. R. (2008), Formal models for expert finding on dblp bibliography data, *in* 'Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)'.

Deng, H., King, I. and Lyu, M. R. (2011), 'Enhanced models for expertise retrieval using community-aware strategies', *IEEE Transactions on Systems, Man, and Cybernetics* .

Dietterich, T. G. (1998), 'Approximate statistical tests for comparing supervised classification learning algorithms', *Neural Computation* **10**(7), 1895–1924.

Duan, H., Zhou, Q., Lu, Z., Jin, O., Bao, S., Cao, Y. and Yu, Y. (2008), Research on enterprise track of trec 2007 at sjtu apex lab, *in* 'Proceedings of the 16th Text REtrieval Conference (TREC 2007)'.

Dwork, C., Kumar, R., Naor, M. and Sivakumar, D. (2001), Rank aggregation revisited, *in* 'Proceeding of the 10th World Wide Web Conference Series'.

Egghe, L. (2006), 'Theory and practise of the g-index', *Scientometrics* **69(1)**, 131–152.

Fang, H. and Zhai, C. (2007), Probabilistic models for expert finding, *in* 'Proceedings of the 29th European Conference on Information Retrieval Research (ECIR '07)'.

Fang, Y., Si, L. and Mathur, A. P. (2010), Discriminative models of integrating document evidence and document-candidate associations for expert search, *in* 'Proceedings of the 33rd International Conference on Research and Development in Infromation Retrieval (SIGIR '10)'.

Farah, M. and Vanderpooten, D. (2007), 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)'.

Fox, E. and Shaw, J. A. (1994), Combination of multiple searches, *in* 'Proceedings of the 2nd Text Retrieval Conference (TREC 1994)'.

Freund, Y., Iyer, R., Schapire, R. E. and Singer, Y. (2003), 'An efficient boosting algorithm for combining preferences', *Journal of Machine Learning Research* **4**, 933–969.

Fu, Y., Yu, W., Li, Y., Liu, Y., Zhang, M. and Ma, S. (2006), Thuir at trec 2005: Enterprise track, *in* ‘Proceedings of the 14th Text REtrieval Conference (TREC 2005)’.

Fuhr, N. (1989), ‘Optimum polynomial retrieval functions based on the probability ranking principle’, *ACM Transactions on Information Systems* **7**, 183–204.

Fuhr, N. (1992), ‘Probabilistic models in information retrieval’, *Computer Journal* **35**, 243–255.

Garfield, E. (2001), From bibliographic coupling to co-citation analysis via algorithmic historio-bibliography, Technical report, Drexel University, Philadelphia.

Gozalo, J., Verdejo, F., Penas, A. and Peters, C. (2000), User needs, Technical report, CLEF.

Haykin, S. (2008), *Neural Networks and Learning Machines*, Pearson Education, 3rd edition.

He, B., Macdonald, C., Ounis, I., Peng, J. and Santos, R. L. (2008), University of glasgow at trec 2008: Experiments in blog, enterprise, and relevance feedback tracks with terrier, *in* ‘Proceedings of the 2008 Text REtrieval Conference (TREC 2008)’.

Hirsch, J. E. (2005), An index to quantify an individual’s scientific research output, *in* ‘Proceedings of the National Academy of Sciences USA (PNAS ’05)’.

Hsu, C.-W., Chang, C.-C. and Lin, C.-J. (2010), A practical guide to support vector classiffication, Technical report, National Taiwan University.

Järvelin, K. and Kekäläinen, J. (2002), ‘Cumulated gain-based evaluation of ir techniques’, *ACM Transactions on Information Systems* **20**, 422–446.

Jena, S., Mendes, C. and Milidiu, R. (2007), Aprendendo o ranking do google utilizando o ranknet, Technical report, Pontifica Universidade Catolica do Rio de Janeiro.

Joachims, T. (2002), Optimizing search engines using clickthrough data, *in* ‘Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD ’02)’.

Joachims, T. (2006), Training linear SVMs in linear time, *in* ‘Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD ’06)’.

Kessler, M. M. (1963), ‘Bibliographic coupling between scientific papers’, *American Documentation* **24**, 123–131.

Kleinberg, J. M. (1999), ‘Authoritative sources in a hyperlinked environment’, *Journal of ACM* **46**, 604–632.

Li, P., Burges, C. and QiangWu (2008), Learning to rank using classification and gradient boosting, *in* ‘Advances in Neural Information Processing Systems (NIPS ’08)’.

Liu, T.-Y. (2009), ‘Learning to rank for information retrieval’, *Foundations of Trends Information Retrieval* **3**, 225–331.

Liu, X., Bollen, J., Nelson, M. L. and de Herbert Van de Sompel (2005), Co-authorship networks in the digital library research community, *in* ‘Information Processing and Management’.

Macdonald, C., Hannah, D. and Ounis, I. (2008), High quality expertise evidence for expert search, *in* ‘Proceedings of the Information Retrieval Research, 30th European conference on Advances in Information Retrieval’.

Macdonald, C. and Ounis, I. (2006), Voting for candidates: adapting data fusion techniques for an expert search task, *in* ‘Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM ’06)’.

Macdonald, C. and Ounis, I. (2007*a*), Expertise drift and query expansion in expert search, *in* ‘Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM ’07)’.

Macdonald, C. and Ounis, I. (2007*b*), Using relevance feedback in expert search, *in* ‘Proceedings of the 29th European Conference on Information Retrieval Research (ECIR ’07)’.

Macdonald, C. and Ounis, I. (2008), Voting techniques for expert search, *in* ‘Knowledge Information Systems’.

Macdonald, C. and Ounis, I. (2011), Learning models for ranking aggregates, *in* ‘Proceedings of the 33rd European Conference on Information Retrieval (ECIR ’11)’.

Manning, C. D. (2008), *Introduction to Information Retrieval*, Cambridge University Press.

Martins, B. and Calado, P. (2010), Learning to rank for geographic information retrieval, *in* ‘Proceedings of the 6th Workshop on Geographic Information Retrieval (GIR ’10)’.

Mattox, D., Maybury, M. T. and Morey, D. (1999), Enterprise expert and knowledge discovery, *in* ‘Proceedings of the 8th International Conference on Human-Computer Interaction: Communication, Cooperation, and Application Design (HCI ’99)’.

Maybury, M. T. (2006), Expert finding systems, Technical report, Center for Integrated Intelligent Systems, Bedford, Massachusetts.

Metzler, D. and Croft, W. B. (2007), ‘Linear feature-based models for information retrieval’, *Information Retrieval* **16(28)**, 1–23.

Montague, M. H. and Aslam, J. A. (2002), Condorcet fusion for improved retrieval, *in* ‘Proceedings of the 11th international conference on information and knowledge management (CIKM ’02)’.

Moreira, C., Calado, P. and Martins, B. (2011), Learning to rank for expert search in digital libraries of academic publications, *in* ‘Progress in Artificial Intelligence’, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp. 431–445.

Moreira, C., Martins, B. and Calado, P. (2011), Using rank aggregation for expert search in academic digital libraries, *in* ‘Simpósio de Informática (INForum ’11)’.

Moreira, C., Mendes, A. C., Coheur, L. and Martins, B. (2011), Towards the rapid development of a natural language understanding module, *in* ‘Intelligent Virtual Agents’, Springer Berlin / Heidelberg.

Nallapati, R. (2004), Discriminative models for information retrieval, *in* ‘Proceedings of the 27th annual international ACM conference on Research and development in information retrieval (SIGIR ’04)’.

Petkova, D. and Croft, B. (2006), Hierarchical language models for expert finding in enterprise corpora, *in* ‘Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI ’06)’.

Petkova, D. and Croft, B. (2007), Proximity-based document representation for named entity retrieval, *in* ‘Proceedings of the 16th ACM conference on Conference on information and knowledge management (CIKM ’07)’.

Ramos, J. (2001), Using tf-idf to determine word relevance in document queries. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.1424.

Riker, W. H. (1988), *Liberalism Against Populism: A Confrontation Between the Theory of Democracy and the Theory of Social Choice*, Waveland Press.

Robertson, S. and Zaragoza, H. (2009), ‘The probabilistic relevance framework: Bm25 and beyond’, *Foundations of Trends Information Retrieval* **3**, 333–389.

Rowe, B. R., Wood, D. W., Link, A. N. and Simoni, D. A. (2010), Economic impact assessment of nist text retrieval conference (trec) program, Technical report, National Institute of Standards and Technology.

Salton, G. and Buckley, C. (1997), *Improving retrieval performance by relevance feedback*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 355–364.

Salton, G., Wong, A. and Yang, C.-S. (1975), ‘A vector space model for automatic indexing’, *Commununity of ACM* **18**, 613–620.

Sanderson, M. (2010), Test collection based evaluation of information retrieval systems, *in* 'Foundations and Trends of Information Retrieval'.

Sebastiani, F. (2003), *Advances in Information Retrieval - 25th European Conference on IR Research, ECIR*, Springer-Verlag.

Serdyukov, P. (2009), Search for Expertise : Going Beyond Direct Evidence, PhD thesis, University of Twente.

Serdyukov, P., Chernov, S. and Nejdl, W. (2007), Enhancing expert search through query modeling, *in* 'Proceedings of the 29th European Conference on Information Retrieval Research (ECIR '07)'.

Serdyukov, P., Rode, H. and Hiemstra, D. (2008), Modeling multi-step relevance propagation for expert finding, *in* 'Proceeding of the 17th ACM conference on Information and knowledge management (CIKM '08)'.

Shen, H., Wang, L., Bi, W., Liu, Y. and Cheng, X. (2008), Research on enterprise track of trec 2008, *in* 'Proceedings of the 17th Text REtrieval Conference (TREC 2008)'.

Sidiropoulos, A., Katsaros, D. and Manolopoulos, Y. (2007), 'Generalized h-index for disclosing latent facts in citation networks', *Scientometrics* **72(2)**, 253–280.

Sidiropoulos, A. and Manolopoulos, Y. (2005), 'A citation-based system to assist prize awarding', *SIGMOD Record* **34**, 54–60.

Sidiropoulos, A. and Manolopoulos, Y. (2006), Generalized comparison of graphbased ranking algorithms for publications and authors, *in* 'Journal for Systems and Software'.

Small, H. (1973), 'Co-citation in the scientific literature: A new measurement of the relationship between two documents', *Journal of the American Society of Information Science* **24**, 265–269.

Smucker, M. D., Allan, J. and Carterette, B. (2007), 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)'.

Sormunen, E. (2002), Liberal relevance criteria of trec - counting on negligible documents, *in* 'Proceedings of the 25th annual international ACM conference on Research and development in information retrieval (SIGIR '02)'.

Sorokina, D., Caruana, R. and Riedewald, M. (2007), Additive groves of regression trees, *in* 'Proceedings of the 18th European Conference on Machine Learning (ECML '07)'.

Taylor, M., Guiver, J., Robertson, S. and Minka, T. (2008), Softrank: optimizing non-smooth rank metrics, *in* ‘Proceedings of the international conference on Web search and web data mining (WSDM ’08)’.

Tsochantaridis, I., Joachims, T., Hofmann, T. and Altun, Y. (2005), ‘Large margin methods for structured and interdependent output variables’, *Journal of Machine Learning Research* **6**, 1453–1484.

Voorhees, E. (1999), The trec-8 question answering track report, *in* ‘Proceedings of Text Retrieval Conference (TREC 1999)’.

Xu, J. and Li, H. (2007), 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)’.

Xu, J., yan Liu, T., Lu, M., Li, H. and ying Ma, W. (2008), 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.

Yang, Z., Tang, J., Wang, B., Guo, J., Li, J. and Chen, S. (2009), Expert2bole: From expert finding to bole search, *in* ‘Proceedings of the 15th ACM Conference on Knowledge Discovery and Data Mining (KDD ’09)’.

Yao, C., Peng, B., He, J. and Yang, Z. (2006), Cnds expert finding system for trec2005, *in* ‘Proceedings of the14th Text REtrieval Conference (TREC 2005)’.

You, G., Lu, Y., Li, G. and Yin, Y. (2007), Ricoh research at trec 2006: Enterprise track, *in* ‘Proceedings of the15th Text REtrieval Conference (TREC 2006)’.

Yue, Y., Finley, T., Radlinski, F. and Joachims, T. (2007), 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)’.

Zhang, J., Ackerman, M. S. and Adamic, L. (2007), Expertise networks in online communities: structure and algorithms, *in* ‘Proceedings of the 16th international conference on World Wide Web (WWW ’07)’.

Zhu, J., Song, D. and Rüger, S. (2007), The open university at trec 2006 enterprise track expert search task, *in* ‘Proceedings of the15th Text REtrieval Conference (TREC 2006)’.

Zhu, J., Song, D. and Rüger, S. (2008*a*), The open university at trec 2007 enterprise track, *in* ‘Proceedings of the 16th Text REtrieval Conference (TREC 2007)’.

Zhu, J., Song, D. and Rüger, S. (2008*b*), The open university at trec 2007 enterprise track, *in* ‘Proceedings of the 16th Text REtrieval Conference (TREC 2007)’.