A Connectionist Model for Adaptive Information Retrieval

Peter Prettenhofer

A Connectionist Model for Adaptive Information Retrieval

Master's Thesis

at

Graz University of Technology

submitted by

Peter Prettenhofer

Knowledge Management Institute (KMI),
Graz University of Technology
A-8010 Graz, Austria

10th December 2008

© Copyright 2008 by Peter Prettenhofer

Advisor: Dr. Stefanie Lindstaedt
Co-Advisor: Dr. Peter Scheir

TU Graz

Ein Konnektionistisches Modell für Adaptives Information Retrieval

Diplomarbeit

an der

Technischen Universität Graz

vorgelegt von

Peter Prettenhofer

Institut für Wissensmanagement (IWM),
Technische Universität Graz
A-8010 Graz

10. Dezember 2008

© Copyright 2008, Peter Prettenhofer

Diese Arbeit ist in englischer Sprache verfasst.

Begutachter:	Dr. Stefanie Lindstaedt
Mitbetreuender Assistent:	Dr. Peter Scheir

Abstract

This thesis explores the use of connectionist models for adaptive information retrieval. In general, adaptive information retrieval refers to the process of adapting the search towards the user's context and need (cf. [Joho et al., 2008]). In particular, the goal of an adaptive information retrieval system is to improve retrieval effectiveness in response to feedback, received on prior performance (cf. [Voorhees, 2008]). The majority of research in this area is dedicated to the adaption to short-term needs by building up a short-term user model. The most prominent short-term adaptive retrieval strategy is the well known *relevance feedback* technique which aims at a better understanding of the user's information need based on relevance data provided by the user. A known limitation of the traditional relevance feedback approach is the fact that it implements a transient form of adaption which is limited to a single query session. According to Jose et al. [2008], the real challenge in adaptive IR, however, is the adaption to long-term needs, interests that evolve over time.

In the course of this thesis a retrieval model for adaptive IR was developed which enables the adaption of the system to such long-term needs by converging ideas from the area of *probabilistic indexing and retrieval* and *neural networks*. The model builds upon the associative retrieval model proposed by Scheir [2008]. The associative model is based on the *spreading activation* paradigm which represents associations among information items (i.e. index terms and documents) as a weighted directed graph, an *associative network*. Using the backpropagation algorithm, the adaptive model modifies the structure of this associative network, based on relevance feedback data provided by the users, to better reflect the user's intuition of relevance.

Furthermore, the thesis describes the reference implementation of the model in the APOSDLE platform and the results of a system evaluation based on a traditional information retrieval evaluation experiment are presented. The experiment reports a significant increase in effectiveness of the adaptive model compared to the original associative retrieval model. The promising results provide preliminary evidence of the benefit of incorporating relevance feedback into the retrieval model proposed by Scheir [2008].

Kurzfassung

Die vorliegende Arbeit untersucht den Nutzen von konnektionistischen Modellen für das Adaptive Information Retrieval. Adaptives Information Retrieval bezeichnet die Anpassung des Suchprozesses an den individuellen Kontext sowie die individuellen Bedürfnisse des Benutzers. Das Ziel von adaptiven Retrieval-Systemen ist, auf Basis von Benutzer-Feedback zur Güte vorheriger Resultate, die Effektivität des Systems zu verbessern. Die Mehrheit der Forschung auf diesem Gebiet betrachtet die Adaption in Bezug auf kurzzeitige Bedürfnisse des Benutzers durch Erstellung von kurzfristigen Benutzermodellen. Die am weitesten verbreitete dieser kurzfristigen Adaptionsstrategien ist die sogenannte *Relevance-Feedback-Technik*. Das Ziel dieser Technik ist, basierend auf Benutzer-Feedback zur Güte (d.h. Relevanz) der gegenwärtigen Resultate, die system-interne Repräsentation des Informationsbedürfnisses des Benutzers zu verbessern. Eine bekannte Einschränkung dieses Ansatzes ist, dass lediglich eine flüchtige Adaption des Systems stattfindet, die auf eine Anfrage-Session begrenzt ist. Laut Jose et al. [2008] stellt die Adaption des Systems in Bezug auf längerfristige Bedürfnisse des Benutzers - also Bedürfnisse und Interessen die sich über einen längeren Zeitram entwickeln - die wahre Herausforderung im Adaptiven IR dar.

Im Rahmen dieser Diplomarbeit wurde ein Modell für das Adaptive Information Retrieval entwickelt, das die Adaption des Systems in Bezug auf langfristige Benutzerbedürfnisse ermöglicht, indem das Modell auf Ideen und Techniken aus dem Bereich der *probabilistischen Indizierung und Retrievals* sowie der *Neuronalen Netze* zurückgreift. Das entwickelte Modell stellt eine Erweiterung des assoziativen Retrieval-Modells von Scheir [2008] dar. Dieses assoziative Modell repräsentiert Assoziationen zwischen Informationsobjekten (d.h. Indexterme und Dokumente) als einen gewichteten Graphen, der auch als *Assoziatives Netz* bezeichnet wird. Die Verarbeitung von Anfragen in Scheiers Modell ist mittels *Aktivierungsausbreitung* analog zur Informationsverarbeitung in Neuronalen Netzen realisiert. Das konnektionistische Modell, das im Laufe der Arbeit entwickelt wurde, benutzt den effizienten Backpropagation Algorithmus, um die Struktur des Assoziativen Netzes - basierend auf Relevanz-Feedback - zu modifizieren, mit dem Ziel, dass das Modell in der Lage ist, den intuitiven Begriff der Relevanz besser wiedergeben zu können.

In weiterer Folge beschreibt die Arbeit die Referenzimplementierung des Modells im APOSDLE-System und präsentiert die Ergebnisse einer Systemevaluierung basierend auf einer kleinen Testkollektion aus Dokumenten, Anfragen und Relevanzbewertungen. Die Evaluierung zeigt eine signifikante Verbesserung der Effektivität des adaptiven Modells gegenüber dem originalen, assoziativen Modell. Somit schafft das vorgestellte Modell die Grundlage, Relevanz-Feedback in das Retrieval-Modell von Scheir [2008] einfließen zu lassen.

I hereby certify that the work presented in this thesis is my own and that work performed by others is appropriately cited.

Ich versichere hiermit, diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Acknowledgements

I am indebted to my former colleagues at the Know-Center and the KMI who have provided a vibrant, charming and fruitful environment, invaluable help and feedback during the course of my work. I would like to thank my advisor, Stefanie Lindstaedt, for giving me the opportunity to work on problems I feel passionate about, for her immediate attention to my questions and endless hours of toil in correcting drafts of this thesis.

I especially wish to thank my co-advisor, Peter Scheir, for his guidance and advice, not only in the course of this thesis but also throughout my master program. He introduced me to the fascinating area of Information Retrieval, that has caught my attention in the past few years, and allowed me to explore it in numerous courses and projects. His passion and enthusiasm about research in general and IR in particular set the course for my future career - thanks Peter!

Lastly, and most importantly, I wish to thank my brother, Toni, my parents, Brigitte and Kurt, and my girlfriend Anna. Without their support and love neither this thesis nor my study would have been possible.

Peter Prettenhofer
Graz, Austria, December 2008

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Chapter 1

Introduction

1.1 Problem Statement

This thesis examines the use of connectionist models to the task of adaptive information retrieval. That is, the automatic adaption of the retrieval system to the user's context and needs in order to better reflect the user's intuition of relevance. Connectionist methods represent established technologies and some of them have already been effectively applied to adaptive information retrieval [Burges et al., 2005; Crestani and van Rijsbergen, 1997; Kwok, 1995; Bartell, 1994; Belew, 1989].

The general goal of this thesis is to complement an existing associative retrieval system based on a spreading activation model with an explicit learning procedure. The existing model, introduced by Scheir [2008], builds on top of an associative network, a weighted graph representing associations among information items (i.e. concepts and documents). Based on explicit feedback provided by the user, the learning procedure should modify the structure of the associative network to optimize the ranking of future queries. An integral part of this thesis is to show, whether or not the application of such a learning procedure is feasible in such a network. If this is possible, the foundations to incorporate relevance feedback in a retrieval system based on the retrieval model proposed by Scheir [2008] could be established.

The fact that the intended learning procedure changes the representation of the network in a *persistent* manner distinguishes the proposed approach from traditional relevance feedback approaches that use feedback information in order to better understand the information need of the user. Rather, the proposed model bears similarities to other methods of parameter learning in information retrieval (IR), namely *probabilistic indexing* [Maron and Kuhns, 1960; Fuhr and Buckley, 1991; Kwok, 1995] and *transformation networks* [Crestani and van Rijsbergen, 1997].

In the course of this thesis the following questions have been addressed:

- Which learning procedure is appropriate for adapting the associative network proposed by Scheir [2008]?
- Based on a system evaluation is the the trained model more effective than the baseline model?

1.2 The APOSDLE Platform

This thesis has been written in the context of the APOSDLE project[1], an EU-funded research project in the area of work-integrated learning. The goal of the research project is to build an advanced process-oriented self-directed learning environment (APOSDLE) which

> *enhances knowledge worker productivity by supporting informal learning activities in the context of knowledge workers' everyday working processes and withing their work environments* [Lindstaedt and Mayer, 2006].

A major milestone of the project was the creation of an infrastructure - referred to as the APOSDLE platform - which supports the integration of working, learning and teaching in an efficient and effective manner to "*provide knowledge workers with optimal guidance to manage the large variety of knowledge artefacts available in the corporate infrastructure*" [Lindstaedt and Mayer, 2006]. The system utilizes domain knowledge about different users, their competencies, tasks and learning needs to proactively assist knowledge workers with relevant *knowledge artefacts* from the corporate document repository. The domain knowledge is stored in a domain ontology, a formal model capturing the necessary aspects of the application domain (cf. [Christl et al., 2008]). The *controlled vocabulary* provided by the ontology is used as a controlled index for classification and retrieval of knowledge artefacts. In the context of APOSDLE, knowledge artefacts refer to documents or parts of documents. In the following, the term knowledge artefact and document are used synonymously.

In this section, a brief introduction to the associative retrieval model proposed by Scheir [2008] is given, which represents the basis of the model that was developed in the course of this thesis. A throughout discussion of the associative retrieval model is deferred to Section 5.1.

For the purpose of retrieval, Scheir [2008] transforms the symbolic representation of the domain ontology into a sub-symbolic representation referred to as an associative network. As pointed out above, an associative network is a directed weighted graph, representing associations among information items. In this context, information items refer to documents and controlled index terms (concepts or *domain model elements*), respectively. Connection weights express the degree of association among these information items. Figure 1.1 shows the topology of such an associative network. From a conceptual point of view, the network comprises two node layers, where the nodes in the first (top) layer represent concepts (i.e. controlled index terms) and the nodes in the second (bottom) layer represent knowledge artefacts (i.e. documents). The inter-layer connections are defined by domain experts in a manual indexing process. However, currently an effort is made in using automated text categorization techniques [Sebastiani, 2002] for the task of automatic controlled indexing. Intra-layer connections and their connection weights are based on heuristics described in Section 5.1. Each learning need is represented by a query comprising a number of concepts. Retrieval of documents is implemented using a *spreading activation* technique: originating from the nodes corresponding to the query concepts, activation flows through the associative network possibly activating document nodes which are not indexed with the initial query concepts. The ranking of the documents is given by the final distribution of activation among the document nodes, where the final activation level of a document node is proportional to the estimated system relevance of the document with respect to the query.

[1] `http://www.aposdle.org` (last visited 24.11.2008)

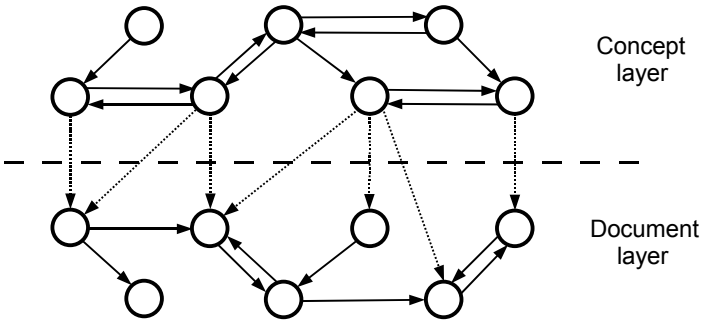

Figure 1.1: The topology of the associative network underlying the associative retrieval model proposed by [Scheir, 2008] from a conceptual point of view. Solid lines indicate intra-layer connections based on heuristics. Dotted lines represent inter-layer connections based on manual indexing. The connection weights which express the degree of association among the items are not shown.

As stated in the previous section the goal of this thesis is to complement this associative retrieval approach with a learning procedure which adjusts the connection weights so that activation (i.e. system relevance) follows some desired pattern. These patterns should be derived from explicit relevance feedback provided by the user.

The reference implementation of the proposed model should be implemented as a component of the APOSDLE platform. The following functional requirements of the model and the reference implementation have been identified:

R1 The adaption of the network should be on-line rather than off-line. The learning procedure should modify the structure of the network immediately after user feedback has been submitted.

R2 The adaption of the network must not be limited to the user who submitted the feedback. This averaging of feedback data across many users enables the construction of a representation of *consensual meaning* of concepts and documents.

R3 The reference implementation of the model should be integrated in the APOSDLE platform.

- **R3.1** A component should be implemented that exposes services to the platform which adapt the structure of the associative network based on explicit relevance judgements with respect to an initial query.
- **R3.2** The reference implementation should build upon the reference implementation of the associative retrieval model proposed by Scheir [2008].
- **R3.3** The modifications of the associative network should be stored in an overlay network. This enables the changing from the original associative model to the connectionist model without the need to rebuild the associative network.

1.3 Methodology

The problem outlined above deals with parameter learning in IR based on relevance feedback data. In general one distinguishes between two fundamental types of learning in IR: (i) *query-focused* and (ii) *document-focused* learning. Query-focused learning is the prevalent learning paradigm in IR and aims at deriving the optimal query representation from the available training data. The well known Rocchio algorithm as well as the traditional probabilistic model of IR, the binary independence model, are instances of this paradigm. Both are described in depth in Chapter 3. Query-focused learning can be regarded as a form of short-term (transient) learning since it is limited to a single query session. Document-focused learning, on the other hand, is a form of long-term learning where the document indexing is modified based on the learning signal. Document-focused learning dates back to the very first probabilistic model of IR, the model of probabilistic indexing [Maron and Kuhns, 1960]. In the 1980s and 1990s various groups examined the use of connectionist representations for retrieval models [Belew, 1989; Kwok, 1995; Crestani and van Rijsbergen, 1997]. In the context of this models, document-focused learning is implemented by changing connection weights in a network. This thesis follows this line of work by using established techniques from the areas of neural networks and pattern recognition - such as the backpropagation algorithm - to implement document-focused learning in the associative network proposed by Scheir [2008].

The proposed extension of the associative retrieval model builds upon the theory of probabilistic indexing [Maron and Kuhns, 1960], which aims to estimate the probability $P(R = 1|d_j, q_i)$ that document d_j is judged relevant given the query q_i by regarding a single document to a number of queries. The theory is implemented by turning the associative network model proposed by Scheir [2008] into a feed-forward neural network. By constraining the activation level of the output layer to the interval $[0, 1]$, the activation can be interpreted as an estimate of the probability of relevance of that document w.r.t. the query. In the course of this thesis only binary relevance was regarded hence relevant and non-relevant documents were expected to have a probability of relevance of 1 and 0, respectively. In this sense, one can think of the network as implementing multiple independent binary classification tasks, one for each document in the collection, where each output unit k gives the posterior probability of document d_k being relevant with respect to the query.

Similar to the binary independence indexing model, presented in Section 3.2.2, the parameters of the model are derived using maximum likelihood by using the cross-entropy error function as the objective function. In order to minimize the error function a simple gradient-based method known as stochastic gradient descent is used. Stochastic gradient descent is an online method that enables the network to be trained one training example at a time rather than a batch of examples. This allows the instant adaption of the system to user feedback rather than having to wait until a sufficient number of learning signals have been aggregated.

The efficient backpropagation algorithm is used to evaluate the gradient of the error function with respect to the network parameters. In addition to this generic learning approach, two variants have been proposed that restrict the adaption of the network to specific weight layers. The variants are based on the observation that the different weight layers exhibit different degrees of sparsity. This sparsity results from poor initialization of connection weights in certain weight layers (cf. [Scheir, 2008]). Retrieval effectiveness might be improved by using as much training data as possible to derive better estimates for these weights.

To evaluate the effectiveness of the proposed extension a traditional IR experiment was

conducted based on a subset of the test collection created by Scheir [2008], containing 1016 documents and 26 queries. The best performing run of the experiment conducted by Scheir was used as the baseline system and the effectiveness of the extensions were evaluated relative to this baseline. Due to the small size of the test collection we used leave-one-out cross-validation to maximize the amount of training data that is used for evaluation. That is for each of the 26 queries the network was trained on all but the query in consideration and the effectiveness of the current query was recorded. After each iteration (i.e. query) the connection weights were reset. The experiment showed that while there is no significant difference between the baseline and the generic approach, both variants significantly outperform the baseline system on the target measure, inferred average precision.

Summing up, in the course of this thesis the following achievements have been made:

- An existing associative retrieval model has been augmented with an explicit learning procedure resulting in a connectionist retrieval model based on a backpropagation neural network.
- The proposed model and two variants thereof have been implemented in the APOSDLE platform.
- The effectiveness of the proposed model and its variants have been evaluated with respect to the original associative model using a traditional IR experiment. The results provide preliminary evidence of the benefit of incorporating relevance feedback into the retrieval model proposed by [Scheir, 2008].

1.4 Structure of the Thesis

Chapter 2 provides a brief introduction to *information retrieval* (IR) by defining the basic concepts of IR. Furthermore, the *relevance feedback* technique is presented which is currently the most prominent paradigm of utilizing relevance information provided by the user. In IR, the term relevance feedback is used in the narrow sense to denote the query-focused learning paradigm mentioned above. Finally, the term *adaptive information retrieval* (AIR) is defined that refers to the adaption of the search process towards the user needs.

In Chapter 3 a broad overview of traditional (supervised) parameter learning approaches based on relevance data is given. The well-known *Rocchio* algorithm to incorporate relevance feedback in the vector-space model is presented. Furthermore, alternative approaches to parameter learning are discussed such as the *probabilistic indexing model* and *description-oriented indexing*. Both aim at deriving probabilistic index term weights based on relevance feedback data.

Chapter 4 begins with a brief introduction to *feed-forward neural network* and the *error-backpropagation algorithm*. Subsequently, various retrieval models based on neural networks are presented, such as *spreading activation models*. The remaining chapter focuses on connectionist approaches to IR.

In Chapter 5 the main contribution of the thesis is presented, an extension to the retrieval model proposed by Scheir [2008] based on backpropagation networks. In the remaining chapter the results of a system evaluation on a small test collection are presented.

Chapter 6 concludes the thesis by summarizing the work and presenting a critical analysis of the results. Furthermore, current trends in the field and opportunities for further research are pointed out.

Chapter 2

Foundations of Information Retrieval

2.1 Information Retrieval

Information retrieval, as an established branch of computer science, deals with the representation, storage, organization of and access to information items [Baeza-Yates and Ribeiro-Neto, 1999, p.1]. The term *information retrieval* (IR) has a very broad meaning and is often loosely-defined. Manning et al. [2008] define information retrieval as "*finding unstructured material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within a large collection (usually stored on computers)*". In general, IR makes no assumptions about the type of information which is retrieved (might be documents, images, audio files, gene expressions). In the course of this thesis, however, we are talking exclusively about textual information (documents) and will use the terms *information retrieval* and *text retrieval* synonymously.
The information retrieval process from a user's perspective proceeds as following:

1. The user generates an *information need*[1]
2. The user encodes her information need into a query (which can be regarded as an incomplete specification of the information need).
3. The system presents a ranked list of documents which matched the query.
4. If the information need is satisfied by the returned documents the process stops. Else, proceed to step two.

Frankly speaking the ultimate goal of an IR system is to retrieve information which might be *relevant* to the user's information need. This is an extremely challenging problem due to several reasons: First, the system has no direct access to the user's information need. Rather, a lossy encoding thereof is presented to the system in the form of a *query*. The loss is subject to various aspects such as the expressiveness of the query language, the skills of the user and the user's knowledge of the collection (i.e. the vocabulary used). Second, the system has to interpret the content of the documents in order to extract syntactic and semantic information from the text. Third, the system has to use this information to determine whether or not a given

[1] Probably due to an anomalous state of knowledge (ASK) [Belkin and Croft, 1992].

	Data Retrieval	Information Retrieval
Matching	Exact match	Partial match, best match
Model	Deterministic	Probabilistic
Query language	Artificial	Natural
Query specification	Complete	Incomplete
Items wanted	Matching	Relevant

Table 2.1: Data versus Information Retrieval (adapted from [van Rijsbergen, 1979]).

document is relevant to a query. Thus, "*the notion of relevance is at the center of information retrieval*" [Baeza-Yates and Ribeiro-Neto, 1999, p.2]. However, relevance is an inherently vague concept, there is still no consensus in the IR community what criteria should be used to determine relevance.

The fact that the nature of the information is unstructured distinguishes information from data retrieval. In data retrieval the matching between the objects and the query is exact - either a tuple matches the query or not. Whereas in information retrieval we usually want to find the *best matching* documents with respect to a query. Furthermore, the query is usually expressed in natural language as opposed to some artificial query language (e.g. SQL). This emphasizes the fact that queries in IR systems are often vague and ambiguous. Table 2.1, adapted from van Rijsbergen [1979], shows the different aspects of data and information retrieval. However, the two columns should not be interpreted as a dichotomy but rather as a spectrum in which IR systems operate. Furthermore, the increasing interest in text retrieval on (semi-)structured data such as XML blurs the boundary between data and information retrieval.

The reminder of this section introduces some basic concepts of an IR system: the task in which the retrieval system operates and the retrieval model which underlies the system. At the end of the section we briefly discuss some practical aspects of the notion of relevance.

2.1.1 Task

We distinguish between two fundamental retrieval tasks: *ad hoc* and *filtering*. In the ad hoc task the document collection indexed by the system is relatively static while new queries are submitted to the system [Baeza-Yates and Ribeiro-Neto, 1999, p.21ff]. A user of such a system is characterized by a short-term information need. This is the kind of operational mode in which web search engines and online public access catalog (OPAC) systems usually operate. In its original form, the ad hoc task does not allow the use of any contextual information of the user (such as location, search history, etc.). Nowadays, web search engines (such as Google[2]) provide such personalized search services, hence they do not operate in the original ad hoc mode.

On the other hand, in the filtering task the users are characterized by a long-term information need (the *user profile*). In a filtering system the set of queries (profiles) is relatively static whereas there is a continuous document stream and the system has to match incoming documents to the user profiles. In information filtering one can further distinguish between: *batch filtering* , *adaptive filtering* and *routing* [Soboroff and Robertson, 2003].

[2] `http://www.google.com`

Batch filtering can be thought of as a binary classification task, whether or not a document matches the profile. Adaptive filtering systems are similar to the former type but in adaptive filtering the user is able to provide feedback to retrieved documents so that the system is able to adapt the users profile based on this feedback. As in information filtering only those documents are presented to the user that match the profile. Hence, there is a sampling bias due to the fact that the user is only able to provide feedback to documents classified as relevant. In the routing task the system generates a ranked list of documents, in contrast to the binary decisions in information and adaptive filtering, respectively. As in adaptive filtering, the user is able to update the profile by providing feedback to documents.

Information filtering is closely related to the area of *text categorization* where documents are assigned to predefined categories. In text categorization the phenomenon of *concept drift* exists which refers to the fact that the explication of a concept (thinking of a category as a concept) may shift over time. For example imagine the concept "Politics", the vocabulary in document instances of this concept may change over time. In 2000 a document containing the terms "Bill" and "Clinton" was likely to be an instance of the category "Politics". However, in 2003 is was more likely for the document to be an instance of a category such as "Welfare" or "Gossip". The phenomenon of concept drift applies analogously to information filtering where the user profile may gradually change over time as the interest of the user does.

2.1.2 Model

As stated in the beginning of the section the ultimate goal of an IR system is to retrieve documents that are relevant to the user's information need. It is the task of an IR model to determine whether or not a document is relevant to the query. More formally speaking, an information retrieval model is a quadruple $\langle \mathbf{D}, \mathbf{Q}, \mathcal{F}, \rho(q_i, d_j) \rangle$ where [Baeza-Yates and Ribeiro-Neto, 1999, p.23f]:

- $\mathbf{D}$ is a set of document representations.
- $\mathbf{Q}$ is a set of query representations.
- $\mathcal{F}$ is a framework for modeling document and query representations and their relationships.
- $\rho(q_i, d_j)$ is the retrieval (or ranking) function that computes a real number for a query representation $q_i \in \mathbf{Q}$ and a document representation $d_j \in \mathbf{D}$. This real number is an estimate of the system relevance of the document given the query.

This formal definition is depicted by the conceptual model for IR introduced by Fuhr [1992] shown in Figure 2.1. It illustrates the relationship between retrieval objects (i.e. documents and queries) and their representations. Following the notation of Fuhr [1992], we use $\underline{q_i} \in \underline{\mathbf{Q}}$ and $\underline{d_j} \in \underline{\mathbf{D}}$ to denote the original documents and queries, respectively. If we consider a set of possible relevance judgements[3] $\mathcal{R} = \{R, \bar{R}\}$, the goal of an IR system is to map each query-document pair to an element of $\mathcal{R}$. Furthermore, there exists an user defined relevance relationship $r : \underline{\mathbf{Q}} \times \underline{\mathbf{D}} \mapsto \mathcal{R}$ that implements this mapping. However, IR systems do not have the required knowledge to directly deal with documents and queries. Rather, they have only

[3] We restrict ourselves here to the case of binary relevance $\mathcal{R} = \{R, \bar{R}\}$ (i.e. either a document is relevant or not). However, the following statements also hold for multi-valued relevance.

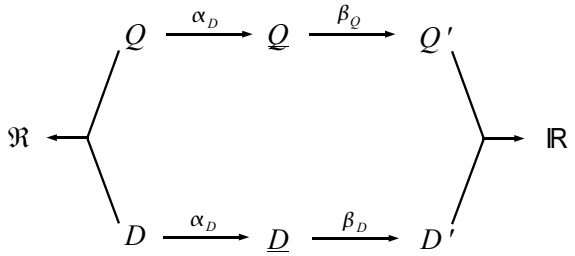

Figure 2.1: A conceptual model for IR (adapted from [Fuhr, 1992]).

a limited understanding thereof based on *representations* of these objects. The mappings α_D and α_Q derive these representations from the original objects. IR indexing models mainly differ in the way these mappings are implemented (cf. [Crestani et al., 1998]). According to Fuhr [1992], "*the representation of an object comprises the data relating to this object that is actually used in the model*". For instance, in the case of full-text indexing the representation (or logical view) of a document might be given by the sequence of tokens in that document (bag-of-words representation) or by the set of terms in that document (set-of-words representation). Fuhr [1992] introduces a further mapping from representations $\mathbf{Q}$ and $\mathbf{D}$ to so-called *descriptions* $\mathbf{Q}'$ and $\mathbf{D}'$. This is implemented by the two mapping functions β_D and β_Q. The sole purpose of this additional step is to enable sophisticated learning models (see Section 3.2.3) that "*aggregate features to allow large enough samples for estimation*" [Crestani et al., 1998]. However, Crestani et al. [1998] points out that most IR models ignore this additional mapping and directly work with the query and document representations. In this case, both β_D and β_Q refer to the identity mapping. Finally, the retrieval function $\rho : \mathbf{Q}' \times \mathbf{D}' \mapsto \mathbb{R}$ is used to compute the retrieval status value (RSV) of each query-document pair. The RSV is an estimate of the system relevance of the document given the query. Subsequently, all documents are ranked w.r.t. the query based on their RSVs.

Concrete IR models differ in the way the mappings α and β as well as the retrieval function ρ are realized. For example, the classical vector-space model of IR uses a bag-of-words representation of documents and queries, respectively, and implements the retrieval function ρ by means of vector similarity in a high dimensional vector space. Since the advent of information retrieval in the 1940s [Bush, 1945] numerous models of IR have been proposed. Among these models, Baeza-Yates and Ribeiro-Neto [1999] distinguish between three top-level categories: (i) boolean (set theoretic), (ii) algebraic and (iii) probabilistic models.

The classical vector-space model discussed in Section 3.1.1 and most of the models based on neural networks presented in Section 4.2 are algebraic models of IR. In algebraic models, documents and queries are represented as vectors in a t-dimensional space (cf. [Baeza-Yates and Ribeiro-Neto, 1999]). Probabilistic models, on the other hand, use probability theory as a framework to handle the uncertainty inherent to IR. Probabilistic models can be subdivided into three broad categories: (a) relevance models which are based on evidence about which documents are relevant to a given query [Crestani et al., 1998], (b) inference models that model the problem of IR as uncertain inference [Pearl, 1988; Turtle and Croft, 1990] and (c) generative models (also known as *language models*) that estimate the probability that a document generated the given query [Ponte and Croft, 1998]. In Section 3.2 we will take a closer look at relevance models. For more information about the classical models of IR consult Manning et al. [2008],

Baeza-Yates and Ribeiro-Neto [1999] or the specific references given above.

2.1.3 The Notion of Relevance

Up to now the concept of relevance was used rather informal. The goal of this section is to elaborate on this inherently vague concept and to introduce a consistent notion of relevance for the rest of the thesis. However, the reader must be warned that despite of the fact that there has been a lot of discussion in the community aiming to "*pin down the concept of relevance*" [Bartell, 1994] to date there has been little consensus. In order to avoid the philosophical pitfalls related to the concept of relevance we will steer the discussion towards the more concrete aspects of relevance - aspects which are indeed crucial for the practical part of this thesis.

According to Saracevic [2007], "*relevance is a, if not even **the**, key notion in information science in general and information retrieval in particular*". Several manifestations of relevance have been recognized. Two important manifestations are the duality: objectivity and subjectivity.

Objective relevance of a document refers to the notions of *topicality* and *aboutness*. According to Bartell [1994] "*a document is objectively relevant to a query if they both refer to a common topic*". Most IR system evaluations are based on objective relevance. It is usually determined based on the inter-annotator agreement of multiple relevance assessors. As a measure for inter-annotator agreement Cohen's kappa can be used (cf. [Manning et al., 2008]).

Subjective relevance on the other hand is the user's perceived utility of the given document. When users provide relevance feedback this is usually an expression of subjective relevance. Subjective relevance encompasses aspects such as topicality, authority and novelty.

Another manifestation of relevance is situational relevance. It refers to the utility of the document with respect to the task that the user is currently pursuing. This manifestation can hardly be captured in practice. It would mean that the fact that a particular document is relevant to a query depends on whether or not the documents that the user has seen before were relevant. This dependency would complicate the automatic determination tremendously. The *Probabilistic Ranking Principle* [Robertson, 1997], one of the few theoretical underpinnings of IR, a principle which guaranties optimal retrieval[4], explicitly assumes that the probability $P(R|q, d_i)$ that a document d_i is relevant to a query q is conditionally independent of the probability $P(R|q, d_j)$ that document d_j is relevant to q. Especially on the web, where duplicate or near-duplicate web pages are a serious problem for search engine vendors, it is obvious that this assumption does not hold (cf. [Manning et al., 2008, p.153]).

System relevance specifies the relevance of a document with respect to a query estimated by an IR system. This estimate is referred to as the *Retrieval Status Value* (RSV).

For an in-depth treatment of the notion of relevance in the context of information science and IR consult Saracevic [2007]. For an overview of different aspects of relevance in information retrieval see Mizzaro [1998].

[4]Given that some assumptions hold and probabilities can be estimated accurately (which they usually cannot)

2.2 Evaluation

We have seen in Section 2.1.2 that a large number of different retrieval models have been proposed. A legitimate question to ask is: given a system based on the retrieval model A is it more *effective* than a system based on a different model B, for a particular retrieval task? Moreover, most of the models comprise a number of adjustable parameters: how do we set these parameters in order to maximize the *effectiveness* of the system? Before we can answer such questions, we have to ask ourselves what it means, for an IR system, to be effective?

This question is not easy to answer. We have already pointed out at the beginning of this chapter that the ultimate goal of an IR system is to satisfy the user's information need by retrieving documents relevant to that need. However, the preceding section showed that the notion of relevance is ill-defined. A holistic evaluation of the retrieval task is difficult - if not impossible - since too many variables influence relevance. Most of these variables are hardly controllable in a laboratory experiment, first and foremost, the user.

In order to free the evaluation experiment "*as far as possible from the contamination of operational variables*" [Cleverdon, 1991] an evaluation methodology was proposed which focuses on the core competency[5] of IR systems. That is, the ability of the system to rank relevant documents before non-relevant documents, for some reasonable definition of relevance . The justification for this abstraction is that if IR systems fail to rank documents properly, it is not likely that they succeed at any real user task (cf. [Voorhees, 2008]). This methodology is known as the *Cranfield paradigm* and, even though its introduction has been highly criticized, it "*has been instrumental in advancing the state of the art in retrieval effectiveness*" [Voorhees, 2008].

The principal tool of the Cranfield paradigm is the *test collection*. A test collection is an abstraction of the retrieval task comprising a predefined set of topics (descriptions of information needs and explications in the form of queries), documents and corresponding relevance assessments. The relevance assessments are based on purely objective judgements of neutral observers. The methodology completely ignores the user involvement in the retrieval task, providing "*more control over the experimental variables at the cost of less realism*" [Voorhees, 2008]. Today, "*test collections are the principal tool used for comparison and evaluation of retrieval systems*" [Sanderson and Zobel, 2005].

To properly evaluate a retrieval system the test collection must be representative for the intended operational setting of the system. The information needs used to evaluate the system should resemble the expected distribution of information needs and ,ideally, the test document corpus should be a subset of the corpus to which the system is finally applied. Test information needs are best designed by domain experts [Manning et al., 2008].

When building a new test collection, the most expensive and time-consuming process is the collection of relevance judgements. Classical test collections such as the Cranfield collection used exhaustive judgements, meaning that the relevance of each query-document pair was assessed. For any non-trivial test collection this is undesirable since too many assessments need to be made. In modern test collections only a subset of the query-document pairs are examined. To determine an adequate subset a method referred to as *pooling* is employed. In pooling, for each query the top k documents retrieved by the different systems are stored in a pool which is then judged exhaustively, whereas different systems include different configurations of a single system.

[5]The definition of the "core competency" of IR systems is due to Karen Spärk Jones (cf. [Voorhees, 2008]).

Given such a test collection comprising explicit relevance judgements, we might define measures of effectiveness that capture the system's ability to rank relevant before non-relevant documents. Again, it has to be noted that the ability of a system to properly rank documents is necessary, tough not sufficient to succeed in satisfying the user's information need (cf. [Voorhees, 2008]).

In order to establish a robust and consistent testbed and benchmark for information retrieval research, the National Institue of Standards and Technology (NIST) promoted an annual conference, dedicated to the evaluation of retrieval systems with large test collections, the Text REtrieval Conference (TREC) [Voorhees and Harman, 2005].

2.2.1 Evaluation Measures

The effectiveness of an IR system is usually evaluated using measures based on recall and precision.

Recall is defined as the number of relevant documents that have been retrieved (true positives) divided by the total number of relevant documents in the collection (true positives plus false positives),

$$Recall = \frac{\#(\text{relevant documents retrieved})}{\#(\text{relevant documents})} \tag{2.1}$$

Precision is defined as the number of relevant documents that have been retrieved (true positives) divided by the total number of retrieved documents (true positives plus false positives),

$$Precision = \frac{\#(\text{relevant documents retrieved})}{\#(\text{retrieved documents})} \tag{2.2}$$

These basic measures have been used to evaluate the effectiveness of boolean retrieval systems. In boolean retrieval the answer to a query is a set of documents rather than a ranking of documents.

In order to evaluate the effectiveness of ranked retrieval systems the basic measures of recall and precision need to be extended. A straight-forward extension is to compute recall and precision given the top k documents retrieved. The effectiveness of the system is then evaluated by plotting precision versus recall for each $k \in \{1, \ldots, N\}$, where N is the total number of documents in the collection. These plots are called precision-recall curves. Precision-recall curves have a distinct sawtooth-like shape (cf. [Manning et al., 2008]) resulting from the fact that the curve sharply drops if a non-relevant document is retrieved (recall stays the same but precision drops). On the other hand, if a relevant document is retrieved both, recall and precision, are increased and the curve moves towards the north-east. Usually it is more useful to smooth the curve by computing interpolated precision values at standard recall levels. Interpolated precision-recall curves are the traditional tool to visualize the effectiveness of a ranked retrieval system [Manning et al., 2008].

AP is defined as the average of the precision value at each relevant document retrieved. That is, considering a query q and the corresponding ranked list of documents $\{d_1, \ldots, d_n\}$, where d_i is ranked higher than d_{i+1}, average precision is given by,

$$AP(q) = \frac{1}{|R|} \sum_{n=1}^{N} P(n) \times rel(n) \tag{2.3}$$

where $|R|$ is the number of relevant documents retrieved, $P(n)$ is the precision value at the top n documents retrieved,

$$P(n) = \frac{\#(\text{relevant docs in top n docs retrieved})}{n} \tag{2.4}$$

and $rel(n)$ is the relevance score of d_n,

$$rel(n) = \begin{cases} 1 & \text{, if } d_n \text{ is relevant} \\ 0 & \text{, otherwise} \end{cases} \tag{2.5}$$

A property of AP, which is typical for most IR evaluation measures, is that it favors rankings that retrieve relevant documents early.

To summarize the effectiveness of a retrieval system across a number of queries usually a score referred to as *mean average precision* (MAP) is used. The MAP score of a retrieval system is the *arithmetic mean* of the average precision values of the individual queries. MAP is the most widely used system-oriented effectiveness measure. It is known to be stable and highly informative. When conducting an IR system evaluation our goal is to choose among a set of system the one that achieves not only the highest but also the most stable performance across all queries. To emphasize stability across all queries some researchers prefer the *geometric mean* over the arithmetic, giving rise to the geometric mean average precision (GMAP) effectiveness measure.

The precision at a specific cut-off level as defined in (2.4) is often used in its own right, for example in evaluating the effectiveness of web search engines. In web search, it is virtually impossible for the user to get a reasonable estimate of recall. Precision at n has the advantage not to rely on an estimate of the size of relevant documents in the collection. Furthermore, using small values of n, drastically reduces the number of documents that need to be judged. However, using precision at n as an effectiveness measure comes with a certain cost: instability.

By using a stable evaluation measure a smaller number of topics is needed to obtain reliable evaluation results (i.e. a ranking of the systems in consideration). Whereas, unstable measures such as precision at 10 need a much larger set of topics to obtain reliable results. When the number of topics is small, stable measures offer a higher discriminative power than unstable measures [Buckley and Voorhees, 2000].

IR evaluation experiments have shown that the variance of effectiveness scores is in general much higher on a per topic basis than the variance on a per system basis. A consequence of this large variability in topic performance is that there has to be a sufficient large number of topics in order to maintain reliability of the evaluation results [Voorhees and Buckley, 2002]. Since the cost for building a test collection is proportional to the size of the topic set, it would be desirable to know the minimum number of topics needed to get reliable results. As a rule of thumb, Buckley and Voorhees [2000] propose a minimum of 50 topics to ensure reliability of results.

2.2.2 Incomplete Judgement Sets

As test collections grow in their size (current test collections contain up to 25 million documents) the number of relevant documents for each query grows too. However, pool size is usually held constant, the traditional TREC pool depth is 50 to 100. As a result only a tiny

fraction (less than 1%) of the retrieved documents are actually judged. Non-judged documents do not have any influence on the MAP score since they are believed to be non-relevant. In order to overcome the problem, one might adapt the pool depth, which increases the annotation cost. An alternative solution would be to increase the pool depth but abandon exhaustive relevance judgements and come up with evaluation measures for incomplete relevance judgement sets.

If average precision is considered to be the holy grail of IR evaluation measures, an evaluation measure is desirable which is not only highly correlated with average precision but also robust with respect to incomplete relevance information. Recently, two such measures have been proposed: binary preference (bpref) [Buckley and Voorhees, 2004] and inferred average precision (infAP) [Yilmaz and Aslam, 2006]. In the following we will focus on infAP since it has been shown to be more robust compared to bpref. Another advantage of infAP compared to bpref is that, when complete judgements are available, infAP and AP are equivalent.

Inferred average precision estimates average precision based on the outcome of a random experiment. Given a pool of which only a sample of the documents are judged, infAP is defined as the average expected precision at the rank of each relevant document in the sample. The expected value of precision at k is given by,

$$E[\text{precision at rank } k] = \frac{1}{k} \cdot 1 + \frac{k-1}{k} \left(\frac{P}{k-1} \cdot \frac{R_k + \epsilon}{R_k + N_k + 2\epsilon} \right) \tag{2.6}$$

where P denotes the pool size, R_k is the number of documents above k known to be relevant and N_k is the number of documents above k known to be non-relevant. To avoid divisions by zero, *Lidstone smoothing* is employed, by which a small ϵ is added to both, R_k and N_k [Yilmaz and Aslam, 2006].

In experiments based on TREC-8 data, Yilmaz and Aslam [2006] showed that, when 30% of the relevance judgements of the depth-100 pool are available, infAP is highly correlated with actual AP. The three scatter plots shown in Figure 2.2 visualize the correlation of infAP and MAP as the judgement set is reduced to 30,10 and 5 percent. On the top of each plot the value of three correlation coefficients is given: root mean squared error (RMS), Kendall's τ and linear correlation coefficient ρ. Even for very small judgement sets, infAP has an RMS error of less than 0.05, showing that infAP is indeed a very close approximation of (mean) average precision.

To date, infAP has been used in a number of retrieval experiments [Büttcher et al., 2006; Soboroff and Craswell, 2006]. Despite of its advantages compared to bpref, the latter has been adopted as the new standard measure for incomplete judgement sets at TREC.

2.3 Relevance Feedback

Relevance feedback is the interactive process of reformulating a query in order to represent the user's information need more accurately, based on partial relevance judgments of documents initially retrieved by the system. Introduced in the mid-1960s, it has proven to be easy to use and unusually effective (in both academic experiments and real-world scenarios) [Witten et al., 1999; Salton and Buckley, 1997].

The basic procedure is as follows:

- The user issues a query q.
- The system presents a ranked list of documents which matched the query.

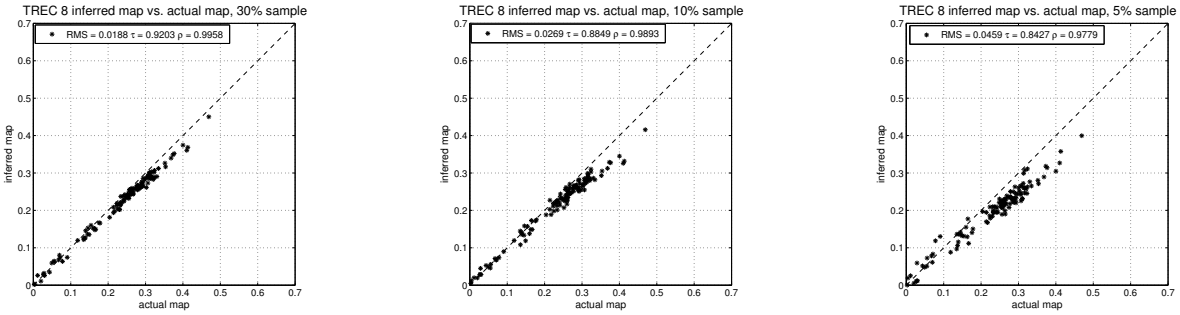

Figure 2.2: Correlation of infAP and MAP based on TREC-8 data. The three scatter plots show the correlation of infAP and MAP as the judgement set is reduced to 30, 10 and 5 percent. Correlation coefficients, root mean squared error (RMS), Kendall's τ and linear correlation coefficient ρ, are shown in the top of each plot. From [Yilmaz and Aslam, 2006].

- The user judges some of the returned documents to be relevant or not.
- The system uses these relevance judgments to create a new query q'.
- The system presents the result of the query q'.

Each retrieval session can be seen as one or more iterations of the procedure described above. The iteration stops when the information need is met or the user abandons the search. The process is based on the assumption that it might be difficult for information seekers to formulate queries when they are not familiar with the vocabulary of a particular collection, but it is easy for users to judge whether or not a particular document meets their information need [Manning et al., 2008, p.174ff].

Salton and Buckley [1997] point out that the main advantages of relevance feedback are:

- It shields the user from the details of the query formulation process, by using relevance judgments to reformulate the initial queries.
- It breaks down the search operation into a sequence of small search steps, making the whole search process more transparent to the user.
- It provides a controlled query alternation process designed to emphasize terms which occur in relevant documents and to de-emphasize terms which occur in non-relevant documents.

For an in-depth and recent discussion of relevance feedback consult the excellent survey by Ruthven and Lalmas [2003].

The relevance feedback process gradually refines a query by applying two basic techniques: *query term weighting* and *query expansion*.

2.3.1 Query Expansion

Query expansion or term selection is the process of selecting new (discriminative) terms from documents judged as relevant and adding them to the query. In the IR literature this is referred to as *local query expansion*. *Global query expansion* on the other hand relies solely on the query terms to find candidate expansion terms. Usually, a thesaurus such as Wordnet[6] is used to find terms associated to the query terms (other global approaches include automatic thesaurus generation at indexing time). We will only consider local methods due to the fact that they "*have shown to be more effective than global techniques in general*" [Xu and Croft, 2000]. For a brief discussion of global methods see Manning et al. [2008, p.173ff].

Query expansion might as well be interactive, which refers to involvement of the user in the term selection process. Ruthven [2003] examines the potential of interactive query expansion.

In the following, we will present two successful approaches to local query expansion:

Term Ranking simply ranks the candidate expansion terms according to their discriminative power to separate relevant from non-relevant documents. Consequently, the top k candidate terms are chosen to expand the query. Estimating the discriminative power of a term is based on the hypothesis that discriminative terms have a different distribution in the relevant documents than in the collection as a whole. In particular, we expect the frequency of good expansion terms to be higher in the set of relevant documents than in the collection. In contrast, terms of mere discriminating power may occur at random with the same frequency in both, the set of relevant documents and the collection. In order to estimate the discriminative power of a term several methods have been proposed. In the following, three approaches selected from [Carpineto et al., 2001] are outlined:

Rocchio's weight Rocchio's algorithm [Rocchio, 1971] is the traditional approach to relevance feedback in the vector-space model. In its original formulation the algorithm does not distinguish between query expansion and query term reweighting. Rather, the query is expanded using all terms from the set of documents judged as relevant. However, experiments [Salton and Buckley, 1997; Harman, 1992] showed that the method is even more effective if only a small number of terms are added to the query, those with the highest indexing weights accumulated over the set of relevant documents. A brief introduction to the vector-space model and a throughout discussion of Rocchio's algorithm is due to Section 3.1. However, in the following the term score as computed by the algorithm is given,

$$score(t) = \sum_{d_i \in R} w(t)_{d_i} \tag{2.7}$$

where $w(t)_{d_i}$ denotes the weight of term t in document d_i.

Robertson Selection Value (RSV) Robertson [1990] showed that, given certain assumptions hold, if index terms are weighted with $w(t)$ they should be selected according to

$$score(t) = \sum_{d_i \in R} w(t)_{d_i}(p_R(t) - p_{N-R}(t)) \tag{2.8}$$

where $p_R(t)$ and $p_{N-R}(t)$ are the probabilities that a relevant and non-relevant document, respectively, contain term t.

[6] http://wordnet.princeton.edu/

Kullback-Leibler divergence Carpineto et al. [2001] proposed an information theoretic approach to term selection. They used the Kullback-Leibler divergence [Manning and Schtze, 1999], an asymmetric measure of the difference of two probability distributions, to score candidate terms. The approach is based on the intuition that good candidate terms contribute more to the divergence of the distribution of terms in the collection p_C from the distribution of terms in the set of relevant documents p_R. The score is given by,

$$score(t) = p_R(t) \cdot \log \frac{p_R(t)}{p_C(t)} \tag{2.9}$$

Local Context Analysis is a local query expansion technique, proposed by Xu and Croft [1996], which selects candidate concepts to augment the query based on cooccurrence with the initial query terms within the top ranked documents. Document concepts are defined as noun groups (i.e. single or consecutive nouns). In contrast to most term selection approaches LCA determines cooccurrence based on document passages instead of documents as a whole. For this purpose, passages (i.e. a text window of fixed size) are extracted from each of the top ranked documents. Local context analysis is a three step process [Baeza-Yates and Ribeiro-Neto, 1999, p.129f]:

- First, the top k passages are retrieved with respect to the initial query.
- Second, for each concept in the top k passages the similarity $sim(q, c)$ between the original query q and the concept c is computed using an adapted tf-idf weighting scheme.
- Third, the top m concepts according to $sim(q, c)$ are added to the original query q. Furthermore, heuristics for weighting the new query terms are applied.

Xu and Croft [2000] report significant performance improvements compared to term frequency based term selection and Rocchio term re-weighting on the TREC-4 and TREC-5 benchmark dataset.

2.3.2 Query Term Weighting

Term weighting is the process of modifying the query term weights in order to emphasize terms which occur in relevant documents and to de-emphasize terms which occur in non-relevant documents. In this sense, query term weighting can be interpreted as a parameter learning problem. Term weighting depends on the retrieval model underlying the system. We will discuss two approaches to query term weighting in Chapter 3: the Rocchio algorithm and the binary independence model.

2.3.3 Pseudo relevance feedback

Pseudo relevance feedback (sometimes called blind relevance feedback) is a technique that provides local analysis without user interaction. The technique simply assumes that the top k documents are relevant and RF proceeds as outlined above. Experiments show that it is very effective in ad hoc system evaluations [Buckley et al., 1994]. However, a poor initial retrieval quality (low precision) might result in even worse performance due to query drift (cf. Section 2.1.1).

2.3.4 Relevance Feedback Evaluation

The effectiveness of pseudo relevance feedback can be evaluated with the standard evaluation approach presented in Section 2.2. However, when evaluating interactive relevance feedback using test collections we face some subtlety. Consider the following example: Given a query q_0, a collection of documents and corresponding relevance judgements. First, we compute the effectiveness of the initial query q_0 on the test collection. To compute the effectiveness of relevance feedback, we reveal the relevance judgements of, for example, the top k documents returned by q_0. This information is used to create the modified query q_1. Again, the effectiveness of q_1 is computed on the whole document collection.

Using such an evaluation strategy is not fair due to the following reason: imagine a simple relevance feedback technique which does not alter the query q_0 but instead only modifies the ranking of the revealed documents (the top k in our example) so that the documents known to be relevant are ranked higher than the document known to be non-relevant. By using such a simple technique, q_1 will always give better or ,at worst, equal effectiveness compared to the initial query q_0.

Using such an evaluation strategy is not fair because it measures the effectiveness on documents that have been used to derive the modified query q_1. A fair evaluation strategy evaluates the system with respect to documents which have not been used to create q_1 [Manning et al., 2008]. Different strategies for evaluating interactive relevance feedback using test collections have been proposed [Chang et al., 1971; Ruthven and Lalmas, 2003], each with their own benefits and drawbacks. In the remainder of this section two popular approaches are described: (a) *residual ranking* and (b) *test and control groups*.

Residual Ranking uses the documents in the *residual collection* - that is, all documents that have not been used for feedback - for the final evaluation of the system. Removing the revealed documents from the test collection might result in inferior performance of the feedback run. In particular, if there are only a few relevant documents in the residual collection. Therefore, the residual ranking method can hardly be used to compare the effectiveness of interactive relevance feedback against a baseline without feedback. However, the method is regularly used to compare different variants of relevance feedback as in [Salton and Buckley, 1997] and the TREC 2008 Interactive Track[7].

Another problem of the residual collection method is that all systems to be evaluated should have the same input documents. If not, the union of all input documents must be removed for the final evaluation because otherwise the different residual collections might exhibit different numbers of relevant documents which makes the evaluation quite problematic.

Test and Control Groups is a method which is based on two document collections. The first collection, which is referred to as the test group, is used to modify the query based on relevance feedback. The retrieval effectiveness of the modified query is evaluated in the second collection - the control group. Running the initial query on the control group enables the direct comparison between the initial and the modified query. The two document collections are usually obtained by randomly splitting a single document collection. However, a random split does not ensure that all relevant documents are equally split among the two collections. Furthermore,

[7] `http://groups.google.com/group/trec-relfeed` (last visited 23.11.2008)

Ruthven and Lalmas [2003] point out that "*neither will it ensure that the relevant documents in the test group are representative of those in the control group*".

2.4 Adaptive Information Retrieval

An emergent subfield of IR is Adaptive Information Retrieval (AIR). Adaptive retrieval is loosely defined as "*a process in which the search is adapted towards the user needs/context*"[Joho et al., 2008]. Hence, it follows that adaptive retrieval is an interactive approach to the task of IR. The introduction of the notion of interactive retrieval might sound redundant because IR itself is inherently an interactive process since an information need is by definition dynamic rather than static. However, in IR it is traditionally modelled as a static entity to ease system evaluation. Adaptive retrieval comprises issues such as context sensitivity, personalization, user modelling and interactive retrieval. A throughout discussion of AIR can be found in [Belew, 2001] and more recent in [Jose et al., 2008].

According to Belew [2001], adaptive retrieval systems "*improve their performance over time, in response to feedback they receive on prior performance*". Adaption can be interpreted in many different ways, it can be related to user's short-term or long-term needs as well as to the user's context or task at hand (cf. [Jose et al., 2008]). Relevance feedback, as presented in the section above, was probably the first technique that implemented an adaptive retrieval strategy. The technique can be viewed as building up a short-term user model. However, the relevance feedback approach is limited since it only provides a *transient* form of adaption limited to the current search session. According to Jose et al. [2008], the real challenge is adapting to long-term needs, interests that evolve over time. One of the earliest attempts to adapt to this long-term needs was due to the SMART group lead by Gerald Salton. Brauen [1969] considered a *document-space modification*, that is to use relevance feedback to "move" the document representation closer to the query representation. This form of adapting the document representation towards the user needs shares many issues with other indexing approaches such as probabilistic indexing and automatic document indexing based on a controlled vocabulary. In the latter a document collection is indexed using a controlled vocabulary by means of automatic text categorization [Sebastiani, 2002]. Based on a manually indexed training set, a hypothesis is formed which is used for the indexing of new documents. However, by providing feedback to whether or not the indexing of a new document was correct, we might update the hypothesis in an *online* fashion - similar to an adaptive librarian.

In the mid/late 1980s, during the renaissance of connectionism, many researchers promoted the use of connectionist representations for adaptive IR, most notably [Belew, 1989; Kwok, 1995; Crestani, 1993; Schütze et al., 1995]. Chapter 4 is dedicated to an in-depth discussion of those attempts. The adaption to long-term user needs requires not only new (adaptive) retrieval models but also new approaches to IR systems evaluation (cf. [Voorhees, 2008]). Today evaluation of adaptive retrieval techniques are mostly limited to simulated evaluations (i.e. user studies) which is not satisfying from a research point of view due to limited reproducibility, high costs and poor generalization.

Chapter 3

Parameter Learning in Information Retrieval

3.1 Relevance Feedback in the Vector Space Model

This section briefly outlines the most widely used retrieval model, the vector-space model (VSM). Subsequently, the Rocchio algorithm is introduced, the first formalization of an relevance feedback algorithm in the vector-space model. The Rocchio algorithm can be interpreted as a parameter learning approach aiming to derive query term weights based on relevance feedback data provided by the user.

3.1.1 The Vector-Space Model

In the *vector-space model* documents are represented as vectors in a high dimensional vector space. Each dimension of the space corresponds to a particular index term. In general, index terms are simply words or word stems. The vector captures the relative importance of the index terms in a document. This notion of relative importance is encoded in a term-weighing scheme which is a heuristic component of the model. In general, the well known tf-idf[1] weighting scheme is used to express the relative importance of the index terms for a document. Term weighting is crucial to the performance of the retrieval system. However, the particular weighing scheme used is insignificant to the discussion that follows. The most popular approaches to term weighting are briefly outlined at the end of this section but the reader is encouraged to consult Manning et al. [2008] for a comprehensive summarization of term weighting.

The dimensions in the term space are orthogonal indicating that the dependencies among the terms in the document are ignored . The document is basically treated as a *bag of words*. Despite of this rather naive representation, which obviously does not give considerations to the latent structure of natural language text, the bag of words representation has been empirically proven to yield good results. It certainly is the common way to represent text in a variety of tasks such as retrieval, categorization and clustering.

In the vector-space model, similarity between documents is expressed by the cosine of the

[1] Term frequency - inverse document frequency.

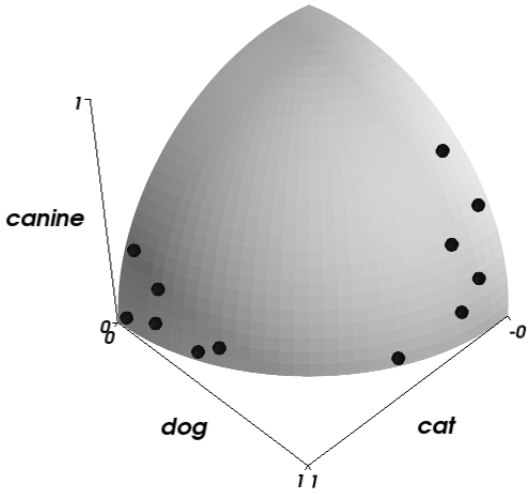

Figure 3.1: Vector-space representation for a collection of documents containing the terms ”dog”, ”cat” and ”canine” (adapted from [Allan, 2007]).

angle of the document vectors. The similarity metric for two document vectors is given by,

$$sim(\vec{d_i}, \vec{d_j}) = \frac{\vec{d_i} \cdot \vec{d_j}}{\|\vec{d_i}\| \, \|\vec{d_j}\|} \tag{3.1}$$

where $\vec{d_i}$ is the vector representation of document i and $\|\vec{d_i}\|$ is the *Euclidean length* of $\vec{d_i}$. The numerator of (3.1) represents the *dot product* (also called *inner product*) of the two vectors, the denominator ensures that the document vectors are length normalized to unit vectors. The dot product of two unit vectors is equal to the cosine of the angle between the two vectors. Each of the length normalized vectors is located on the surface of the unit (hyper -) sphere in the vector space (cf. [Manning et al., 2008, p.290f]). Figure 3.1 illustrates this unit sphere for a collection of documents (represented by the dark dots) containing the index terms ”dog”, ”cat” and ”canine”. The greater the distance between two documents on the surface of the (hyper -) sphere, the greater is the cosine of the angle between the two documents. The vector-space model does not allow negative term weights. Hence, the document vectors are always located on the surface of the positive quadrant of the (hyper-) sphere.

By viewing a query as a document one can compute the similarity between a query $\vec{q}$ and a document $\vec{d_j}$ simply by computing the cosine of the angle of the two vectors analogue to what is described above. The similarity is given by,

$$sim(\vec{q}, \vec{d_j}) = \frac{\vec{q} \cdot \vec{d_j}}{\|\vec{q}\| \, \|\vec{d_j}\|} \tag{3.2}$$

where $\vec{q}$ is the vector representation of the query. Figure 3.2 shows the same ”dog”, ”cat” and ”canine” example, where the green dot represents a query containing the term ”canine”. The blue isocline indicate the distance on the unit sphere with respect to the query vector $\vec{q}$.

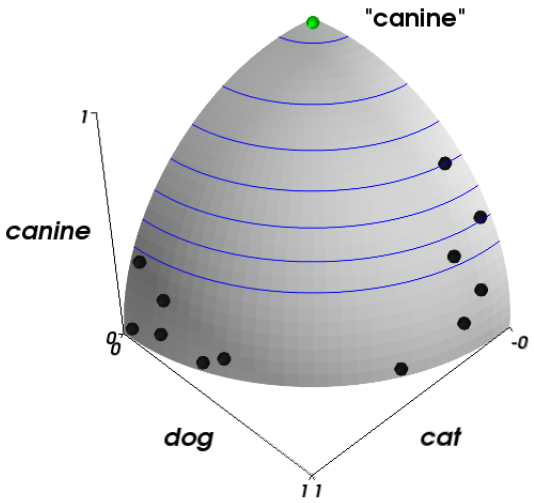

Figure 3.2: Vector-space representation for a collection of documents containing the terms ”dog”, ”cat” and ”canine” and a query (green dot) for the term ”canine”. The blue isocline show the distance from the query on the unit sphere (adapted from [Allan, 2007]).

Using the cosine similarity as the scoring function we can produce a ranking given a query $\vec{q}$ simply by computing the cosine similarity between the query and each document in the collection and rank the documents by decreasing score. Again considering the example in Figure 3.2, the highest ranked documents with respect to the query ”canine” are the two upper right documents (the so speaking ”dogish caninish” documents) which have the highest similarity to the query vector and hence, the smallest distance to the query on the unit sphere (as can be seen by the isocline).

In contrast to other retrieval models, the vector-space model does not explicitly attempt to predict whether or not a document is relevant but rather ranks the documents based on their *degree of similarity* to the query [Baeza-Yates and Ribeiro-Neto, 1999; Manning et al., 2008].

Term-weighting schemes As described above, the term-weighting scheme is a major heuristic component of the model. We will briefly describe the most successful weighting scheme, the tf-idf weighting scheme, and refer to Manning et al. [2008, p.107ff] for an in-depth discussion of term weighting and alternative weighting schemes. The tf-idf weighting scheme assigns each term t in document d a numerical weight by combining two statistical properties of the term t and document d: (a) the *term frequency* of a term t in a document d, denoted by $tf_{t,d}$, and (b) the *inverse document frequency* (idf) of a term t. The assumption of using raw term frequency as an indicator of relevance is that the number of occurrences of a term in a document states something about the ability of term t to capture the content of document d. Raw term frequency suffers from the fact that it considers all terms equally important. However, different terms have different discriminating power when assessing relevance. Some terms, especially *stop words* (e.g. ”the”, ”a”, ”is”), tend to occur throughout the document collection. Hence,

they are of mere discriminating power. This is captured by the inverse document frequency, which considers the discriminating power of an individual term by assuming that terms which only occur in a small number of documents are more discriminative than terms which appear in a large number of documents. The inverse document frequency of a term t is defined as follows:

$$idf_t = \log \frac{N}{df_t} \tag{3.3}$$

where N is the total number of documents in the collection and df_t is the number of documents which contain term t. Combining the two properties leads to the following definition of the tf-idf weight of a term t in document d:

$$\text{tf-idf}_{t,d} = tf_{t,d} \times idf_t \tag{3.4}$$

The highest tf-idf score is assigned to term t within document d if it occurs many times in d but only within a small number of documents. On the other hand, the tf-idf score is lower if term t occurs within many documents or occurs fewer times in d.

Document length normalization Another major heuristic component of the VSM, which is critical to the performance of the system, is the document length normalization. We have actually seen an instantiation of document length normalization in the cosine similarity measure (3.2), where we normalized each term vector by its Euclidean length turning them into unit vectors. Singhal et al. [1996] introduced another document length normalization method yielding very good experimental results. It addresses the following subtleties of longer documents: (1) longer documents contain more distinct terms and (2) longer documents have higher tf scores.

Singhal et al. [1996] observed that long documents can be grouped into two broad categories: (a) documents that basically repeat the same content over and over again, but where the length of the document does not influence the relative weight of each term and (b) documents which can be considered as a mixture of different topics and where the query terms match only a small segment of that document. In the last case the relative weights of the terms are different from a single short document that matches the query terms.

To address this Singhal et al. [1996] proposed a normalization that is independent of term and document frequency. The proposed document length normalization is based on the following observation: Given a test collection containing a set of documents, queries and a set of relevance judgements indicating whether or not a given document is relevant to a particular query. One can compute the *probability of relevance* of a document as a function of document length, averaged over all queries in the collection [Manning et al., 2008, p.118]. By binning the documents by their length into a number of equally sized bins, the probability of relevance for each bin can computed by simply dividing the number of relevant documents in each bin by the total number of documents in the corresponding bin. By plotting the probability of relevance against the median document length of each bin we might obtain a curve which is similar to the thick curve in the left plot of Figure 3.1.1. If we would use system relevance, as computed by cosine similarity (3.2) with Euclidean length normalization, instead of the probability of relevance we might obtain a curve similar to the thin curve. The plot shows that Euclidean length normalization has a tendency to over-estimate the system relevance of shorter documents and to under-estimate the relevance of longer documents (cf. [Singhal et al., 1996]). Both curves intersect at a point p which corresponds to the document length l_p. This length is referred to as the *pivot length*. Based on this observation Singhal et al. [1996] propose to fix the Euclidean

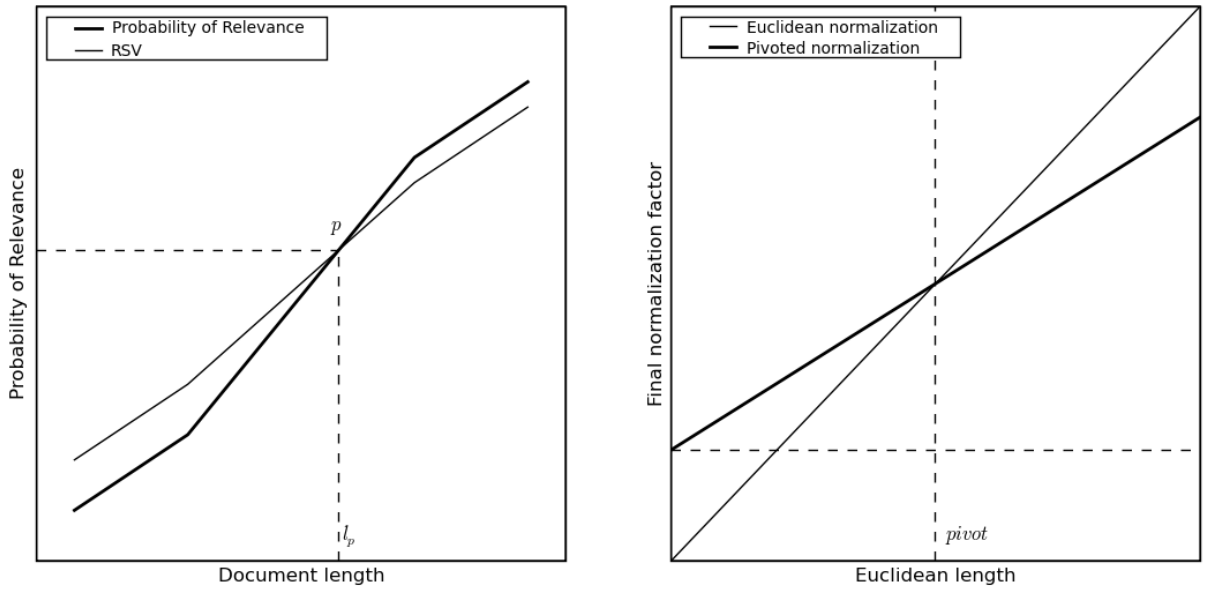

Figure 3.3: Pivoted document length normalization: (a) Cosine normalization distorts system relevance (thin) compared to probability of relevance (thick) at the expense of longer documents. (b) Implementing pivoted document length normalization by linear scaling. (adapted from [Manning et al., 2008, p.118] and [Singhal et al., 1996]).

normalization curve at point p and rotate (*tilting*) it counter-clockwise in order to match the thick curve more closely. The resulting document length normalization creates document term vectors that are not necessarily of unit length. For documents that are less than l_p we normalize more than Euclidean length. For documents larger than l_p we normalize less than the Euclidean length of that document. By computing the dot product of the unit length query term vector and the normalized document term vector we obtain a score which accounts for the effect of document length on relevance [Manning et al., 2008, p.118]. The idea is illustrated in Figure 3.1.1 (b), the Euclidean length $\|\vec{d}\|$ of a document $\vec{d}$ is represented on the x-axes and the final length normalization factor is shown on the y-axes. The normalization performed by the cosine similarity (3.2) is represented by the thin line ($y = x$). A simple form of pivoted document length normalization, depicted by the thick line, uses a normalization factor that is linear in $\|\vec{d}\|$ but with a *slope* $a < 1$. The slope a (i.e. the amount of rotation) as well as the Euclidean length $pivot$ where the two lines intersect are the parameters of the term weighting model:

$$n_{piv}(\vec{d}) = a\|\vec{d}\| + (1 - a)pivot, \tag{3.5}$$

3.1.2 The Rocchio Algorithm

Rocchio [1971] interprets the problem of retrieval as that of defining the optimal representation of the user's information need. The Rocchio algorithm defines an optimal query by maximizing the difference between the average term vector of the relevant document and the non-relevant documents, respectively. The assumption of the Rocchio algorithm is that documents identified as relevant share common properties (have similarities among themselves) and documents iden-

tified as non-relevant have properties which are dissimilar to those of the relevant documents. The basic idea of the algorithm is to move the query vector to the centroid of the cluster of relevant documents and away from the centroid of the cluster of non-relevant documents.

Formally, the optimal query vector $\vec{q}_{opt}$ is given by,

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\forall \vec{d}_j \in C_r} \vec{d}_j \; - \; \frac{1}{N - |C_r|} \sum_{\forall \vec{d}_j \notin C_r} \vec{d}_j \tag{3.6}$$

where,

C_r is the set of all relevant documents in the collection;

$|C_r|$ is the number of relevant documents in the collection;

N is the set of all documents in the collection;

$\vec{d}$ is the term vector of a particular document d.

However, due to the fact that the set of relevant documents C_r is in general not known a priori, the optimal query $\vec{q}_{opt}$ has to be approximated. This can be done by formulating an initial query and to incrementally modify this query in order to approximate the optimal query. The reformulation is done by considering only those documents that are known (according to user judgments) to be relevant or non-relevant at that point of time. This yields the classical Rocchio algorithm to calculate the modified query $\vec{q}_m$,

$$\vec{q}_m = \alpha \, \vec{q}_0 + \beta \, \frac{1}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j \; - \; \gamma \, \frac{1}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \tag{3.7}$$

where D_r and D_n are the set of relevant and non-relevant documents respectively, $\vec{q}_0$ is the initial query and α, β, γ are tuning constants. The tuning constants control the balance between the initial query and the feedback as well as between documents judged as relevant and non-relevant. For example, if the number of judged documents is rather small we want α to be much higher than β and γ because otherwise the modified query would be too biased. According to empirical studies positive feedback turns out to be much more valuable than negative feedback. Therefore, $\beta > \gamma$ is used in general. Typically, the term weights $\hat{q}_{m,i}$ for $i \in 0, \ldots, V$ of the modified query vector $\vec{q}_m$ are restricted to be non negative.

$$\hat{q}_{m,i} = \begin{cases} q_{m,i} & \text{if } \; q_{m,i} > 0 \\ 0 & \text{else} \end{cases} \tag{3.8}$$

where $q_{m,i}$ represents the i-th term component of the query vector $\vec{q}_m$.

Equation (3.9) refers to a variant of the Rocchio algorithm which takes only the highest ranked non-relevant document into account. Salton and Buckley [1997] found that this variant, called *Ide dec-hi*, achieves the most consistent performance compared to other variants of the algorithm.

$$\vec{q}_m = \vec{q}_0 + \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j \; - \; max_{non-relevant}(\vec{d}_j) \tag{3.9}$$

Figure 3.4 illustrates the procedure based on the ”dog”, ”cat”, ”canine” example presented in Section 3.1.1. Given that the initial query is ”canine” and the user indicates that the red

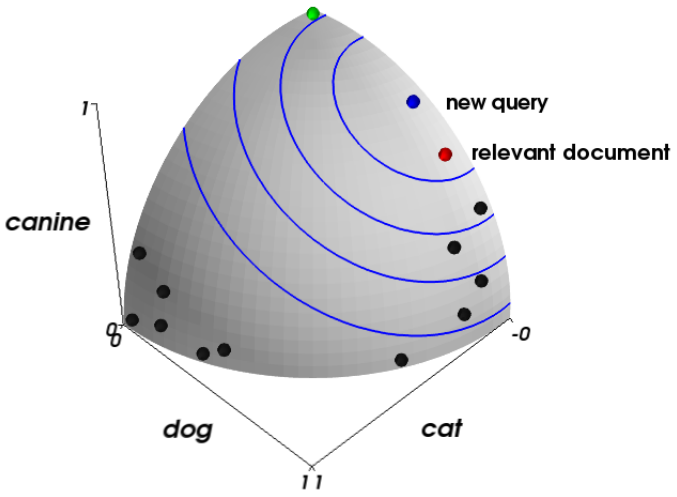

Figure 3.4: Illustration of Rocchio's algorithm for the "dog", "cat", "canine" example. The green dot represents a query for the term "canine", the red dot is a document identified as relevant by the user and the blue dot indicates the modified query created by Rocchio's algorithm (adapted from [Allan, 2007]).

Method	CACM	CISI	CRAN	INSPEC	MED
Baseline	0.145	0.118	0.115	0.136	0.334
Rocchio ($\beta = 0.75$ and $\gamma = 0.25$)	0.255 (+75%)	0.140 (+19%)	0.295 (+156%)	0.182 (+33%)	0.563 (+68%)
Ide dec hi	0.270 (+86%)	0.174 (+47%)	0.301 (+160%)	0.214 (+56%)	0.630 (+88%)

Table 3.1: Experimental results of Rocchio and "Ide dec hi" on five classical test collections (adapted from [Salton and Buckley, 1997]).

document is relevant (a "dogish" document), the algorithm creates a modified query according to (3.7) by interpolating between the original query vector and relevant document (which is the centroid of the relevant documents). The modified query vector $\vec{q}_m$ is represented by the blue dot and the distance from this vector on the surface of the sphere is again shown by the blue isocline. The feedback procedure shifted the query vector to the documents in the lower right of Figure 3.4.

Table 3.1 summarizes some experimental results presented by Salton and Buckley [1997]. The experiment analyzes the effectiveness of the Rocchio algorithm and the "Ide dec hi" variant on a number of test collections. The table shows the effectiveness in terms of average precision (at three particular recall points of 0.75, 0.50, 0.25) as well as the percentage improvement on the baseline. The evaluation was performed using the *residual collection* method. This method simply removes the relevant and non-relevant documents used for feedback and determines the performance on the residual collection. The method maintains the correct relative difference

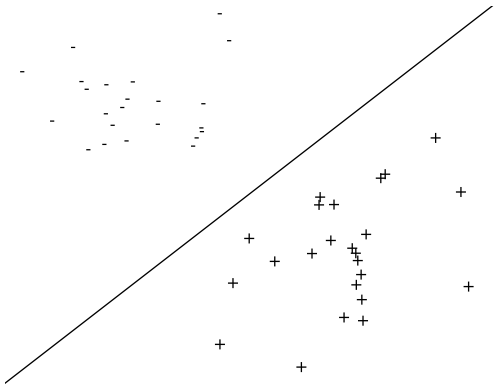

Figure 3.5: A query as a linear discriminator separating the relevant (positive) from the non-relevant (negative) documents.

.

between the initial (the baseline; without feedback information) and the feedback runs, while depressing the absolute performance level in terms of precision and recall. The top 15 documents returned by the initial search were used to construct the feedback vector. For more information on the experimental setup consult Salton and Buckley [1997]; see Baeza-Yates and Ribeiro-Neto [1999, p.91ff] for a brief outline of the CACM, CISI and MEDLINE test collections.

A number of techniques have been proposed to further enhance the parameter estimates (query term weights) produced by Rocchio's algorithm, among which (a) *query zoning* Singhal et al. [1997] and *dynamic feedback optimization* Buckley and Salton [1995] are the most effective.

3.1.3 Connections to Supervised Machine Learning

According to Lewis and Jones [1996] a query can be interpreted as a linear classifier,

$$f(x) = \vec{w} \cdot \vec{x} = \sum_{i=1}^{M} w_i x_i \tag{3.10}$$

where $\vec{w} = \frac{\vec{q}}{\|\vec{q}\|}$ is the normalized query and $\vec{x} = \frac{\vec{d}}{\|\vec{d}\|}$ the normalized document representation. Hence, a normalized query vector represents a linear decision boundary in the vector space. In this sense, relevance feedback can be cast as a supervised machine learning task. The goal of RF is to learn the weights $\vec{w}$ that separate the relevant from the non-relevant documents. If the documents are linear separable, as illustrated in Figure 3.5, a learning algorithm can be chosen that is guaranteed to converge to the optimal query vector $\vec{w}$. If the documents are not linear separable, a learning algorithm can be chosen which minimizes the amount of error [Allan, 2007]. The Rocchio algorithm is just one among many algorithms that can be used for this purpose (cf. Lewis and Jones [1996]).

Hull [1994] was among the first to apply the Rocchio algorithm to the task of automated text categorization. Joachims [1997] gives a probabilistic interpretation of the heuristic components

of the Rocchio algorithm. Schapire et al. [1998] have found that Rocchio classifiers together with query zoning and dynamic feedback optimization can be competitive in terms of effectiveness with state-of-the-art machine learning methods such as Boosting while being much quicker to train and more robust when trained with only few examples. Sebastiani [2002] surveys the use of Rocchio classifiers in automated text categorization.

3.2 Relevance Models

Relevance models, a subclass of the probabilistic models of IR, are based on evidence about which documents are relevant to a given query (cf. [Crestani et al., 1998]). Relevance models capture the notion of relevance in a random variable R. In the following section we regard R as a binary random variable, either relevant $R = 1$ or non-relevant $R = 0$. However, most of what is presented in the course of this section extends to ordinal relevance scores. In a sense, relevance models can be thought of as probabilistic classification models trying to estimate the class conditional probability $P(R = 1|q, d)$, that is the probability that document d will be judged relevant given query q. This section presents a number of approaches to fit such a model based on relevance data provided by the user. The theoretical foundation of relevance models is a decision theoretic concept known as the *probabilistic ranking principle* (PRP) .

> *If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.* [van Rijsbergen, 1979]

Proven by Robertson [1997], the PRP guarantees optimal retrieval when documents (i.e. their representations) are ranked according to their *probability of relevance*, that is the probability of being judged relevant to the query. We will denote the probability of a document representation $\vec{d}$ to be judged relevant with respect to a query representation $\vec{q}$ by $P(R = 1|\vec{d}, \vec{q})$. The PRP makes two fundamental assumptions: (1) the relevance $P(R = 1|\vec{d_i}, \vec{q})$ of a document representation $\vec{d_i}$ with respect to a query representation $\vec{q}$ is independent of the relevance $P(R = 1|\vec{d_j}, \vec{q})$ of a document representation $\vec{d_j}$ w.r.t. $\vec{q}\ \forall j \neq i$ and (2) the probability $P(R = 1|\vec{d}, \vec{q})$ can be estimated accurately.

Of course, the acceptability of the assumptions can be questioned. Assumption 1 is certainly not desirable, since it allows the system to return duplicate documents (cf. [Manning et al., 2008, p.205]). In practice, assumption 2 cannot be satisfied since too many variables are involved in the representation of documents compared with the small amount of document relevance information available (cf. [Crestani et al., 1998]).

Relevance model differ primarily in the way they use this relevance information in order to estimate (i.e. to learn) those probabilities. According to Fuhr [1992], "*if we regard probabilistic IR models as (parameter) learning methods, then three different approaches [...] can be distinguished*":

- query-related learning
- document-related learning

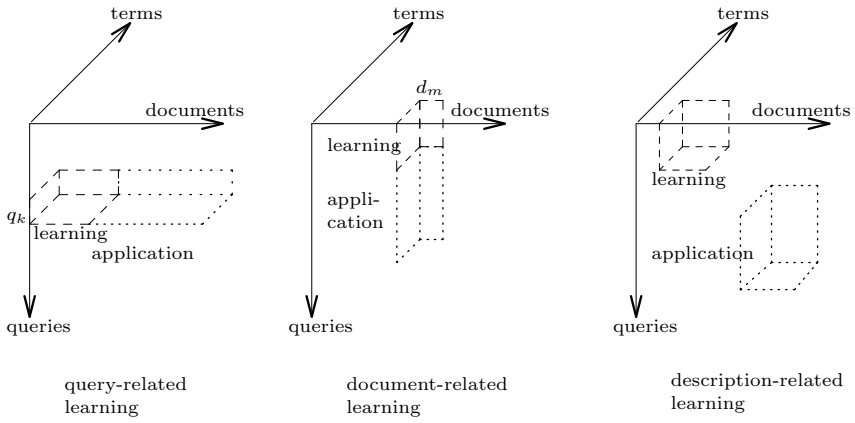

Figure 3.6: Learning approaches in IR (from [Fuhr, 1992])

- description-related learning

The three different approaches are shown in Figure 3.6. Each approach is embedded in a three dimensional space, where the axis refer to the different objects probabilistic parameter relate to (i.e. document, terms and queries). In each of the approaches Fuhr [1992] distinguishes between a learning phase and an application phase. In the learning phase we are presented a subset of documents, terms and queries $Q_L \times D_L \times T_L$ of $Q \times D \times T$ (where Q, D and T are the set of queries, documents and term, respectively) along with corresponding relevance judgements. This relevance feedback information is used to derive probabilistic parameters. In the application phase these parameters are used to improve the descriptions of documents and queries [Fuhr, 1992].

In the query-related learning approach, relevance feedback data is used in the learning phase to derive probabilistic query terms weights for a given query. In the application phase, those weights can be used to rank all the documents in D but are restricted to the single query. Although not a probabilistic model, the Rocchio algorithm introduced in Section 3.1.2 can be regarded as a query-related learning approach. Another instance of this learning approach, the *binary independence model* (BIM), will be discussed in Section 3.2.1. Query-related learning actually refers to the technique known as relevance feedback introduced in Section 2.3. The learning paradigm is suitable, in particular, for the routing task[2] but it makes poor use of the relevance feedback information for ad-hoc information retrieval since the derived parameters are restricted to a single query.

In document-related learning, probabilistic parameters are derived for a single document d_m based on relevance feedback data from a set of terms T_L occurring in a set of queries Q_L. The derived parameters can only be used to rank the single document d_m but with respect to all queries in Q involving terms from T_L. In IR, models employing such a learning approach to derive probabilistic document term weights are known as *probabilistic indexing models*. However, Fuhr [1992] pointed out that "*the major problem with this approach, [however], is the fact that there are not enough relevance judgements for a single document in real databases, so it is almost impossible to estimate the parameters of this approach*". A concrete probabilistic index-

[2]See Section 2.1.1 for a brief discussion of the routing task.

ing model, the *binary independence indexing* (BII) model is briefly discussed in Section 3.2.2. Another probabilistic indexing model, which is based on neural networks [Kwok, 1995], is discussed in Section 4.2.2. The retrieval model which has been developed in the course of this thesis is also inspired by ideas from probabilistic indexing.

Description-related learning overcomes the deficiencies of the document-related learning approach by introducing the concept of (relevance) descriptions. For each term-document pair a relevance description is derived, consisting of features referring to properties of the term, the document and their relationship, similar to the concept of feature vectors in pattern recognition. In the learning phase, parameters for those properties are derived which are used to map from relevance descriptions to probabilistic index term weights. In the application phase, there are no restrictions on the subset of documents, terms and queries, to which the parameters can be applied. Hence, description oriented indexing make the best possible use of the relevance feedback data available. The first indexing approach implementing description-related learning was the *Darmstadt Indexing Approach* (DIA), it is shortly discussed in Section 3.2.3.

3.2.1 The Binary Independence Model

The binary independence model, proposed by Robertson and Jones [1976], is the traditional probabilistic model satisfying the PRP . "Binary" refers to the fact that the documents are represented as binary term vectors. Either a term is present in a document or not. The model does not take into account within document term frequency. "Independence" refers to the modeling assumption that the presence/absence of a term in a document/query is independent of the presence/absence of the other terms. This is essentially the same as the orthogonality assumption used in the vector-space model presented in Section 3.1.1. Its sole purpose is to make the parameter estimation computationally tractable. The subsequent formal definition of the model follows Manning et al. [2008, p.204ff].

The BIM aims to rank the documents in the collection in decreasing order of their probability of relevance $P(R = 1|\vec{d}, \vec{q})$. By applying Bayes theorem we get,

$$P(R = 1|\vec{d}, \vec{q}) = \frac{P(\vec{d}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{d}|\vec{q})} \tag{3.11}$$

However, the model does not estimate this probabilities directly, rather we estimate the odds of the probability of relevance. The odds of an event is a monotonic function hence the document ranking does not change but this eases subsequent computations. The odds of an event R is given by,

$$O(R) = \frac{P(R)}{P(\bar{R})} \tag{3.12}$$

A plot of the odds function is shown in Figure 3.7. Substituting (3.11) into (3.12) gives

$$O(R|\vec{d}, \vec{q}) = \frac{P(R = 1|\vec{d}, \vec{q})}{P(R = 0|\vec{d}, \vec{q})} = \frac{\frac{P(\vec{d}|R=1,\vec{q})P(R=1|\vec{q})}{P(\vec{d}|\vec{q})}}{\frac{P(\vec{d}|R=0,\vec{q})P(R=0|\vec{q})}{P(\vec{d}|\vec{q})}} = \frac{P(R = 1|\vec{q})}{P(R = 0|\vec{q})} \cdot \frac{P(\vec{d}|R = 1, \vec{q})}{P(\vec{d}|R = 0, \vec{q})} \tag{3.13}$$

Note that the common denominator in (3.11) can be ignored, which was the intention to use the odds instead of the probability of relevance directly. The left part of the rightmost expression of (3.13) is constant for all documents given a particular query. Since it does not affect the ranking

it can be ignored. Evaluating the right part of the rightmost expression requires to estimate the joint probability of the components of the document representation (i.e. one random variable per term). This joint probability is extremely difficult to compute because of the complex dependence structures of the random variables. In order to make this computation practicable the term dependence structure is completely ignored hence the term random variables are assumed to be conditionally independent. This is known as the *Naive Bayes conditional independence assumption*. Assuming conditional independence,

$$O(R|\vec{d},\vec{q}) \propto \frac{P(\vec{d}|R=1,\vec{q})}{P(\vec{d}|R=0,\vec{q})} = \frac{P(d_1,\ldots,d_M|R=1,\vec{q})}{P(d_1,\ldots,d_M|R=0,\vec{q})} = \prod_{t=1}^{M} \frac{P(d_t|R=1,\vec{q})}{P(d_t|R=0,\vec{q})} \tag{3.14}$$

where M is the size of the vocabulary. Since the quantities d_t are binary $d_t \in \{0,1\}$ (i.e. the term is present or not) the product in (3.14) can be split into two separate products, one for the terms that appear in the given document and one for those that do not.

$$O(R|\vec{d},\vec{q}) \propto \prod_{t:d_t=1} \frac{P(d_t=1|R=1,\vec{q})}{P(d_t=1|R=0,\vec{q})} \cdot \prod_{t:d_t=0} \frac{P(d_t=0|R=1,\vec{q})}{P(d_t=0|R=0,\vec{q})} \tag{3.15}$$

Consequently, $p_t = P(d_t=1|R=1,\vec{q})$ is used to denote the probability that a term appears in a relevant document given the query and $u_t = P(d_t=1|R=0,\vec{q})$ is used to denote the probability that a term appears in a non-relevant document. Since, $P(d_t=1|R=1,\vec{q})+P(d_t=0|R=1,\vec{q})=1$ it follows that $P(d_t=0|R=1,\vec{q})=1-p_t$. Table 3.2 shows these probabilities in a contingency table.

	document	relevant ($R=1$)	non-relevant ($R=0$)
term present	$d_t=1$	p_t	u_t
term absent	$d_t=0$	$1-p_t$	$1-u_t$

Table 3.2: Contingency table for the BIM; Columns sum to 1. (adapted from [Manning et al., 2008]).

Remember, that the products in (3.15) are defined over all terms. Rather, it would be much more convenient (and efficient) to consider only those terms that appear in the query[3]. By making the assumption that terms that are not present in the query are equal likely to appear in relevant and non-relevant documents, that is $p_t = u_t$ if $q_t = 0$, we can change the products in (3.15) to only consider terms which are present in the query,

$$O(R|\vec{d},\vec{q}) \propto \prod_{t:d_t=1,q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:d_t=0,q_t=1} \frac{1-p_t}{1-u_t} \tag{3.16}$$

The left product is over the query terms that appear in the document whereas the right product is over the query terms not appearing in that document (cf. [Manning et al., 2008]).

Next (3.16) is manipulated by changing the right product to be defined over all query terms, whether they are present or not. At the same time, we amount for the included terms in the left

[3] As implicitly done in the cosine similarity measure (3.1) in the vector-space model.

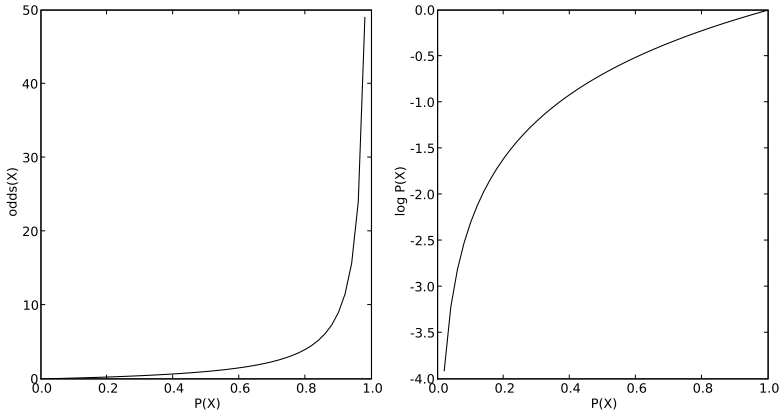

Figure 3.7: Plots of two monotonic functions: (a) the odds of an event X and (b) the (natural) logarithm.

product by dividing through them, so that the value of the expression is unchanged,

$$O(R|\vec{d},\vec{q}) \propto \prod_{t:d_t=1,q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t} \propto \prod_{t:d_t=1,q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \tag{3.17}$$

Again, we see that the right part of the expression is constant given a particular query hence it can be ignored. In order to determine the system relevance (RSV) of a document given a query we simply need to evaluate (3.17). However, the evaluation requires the product over a number of very small quantities, possibly resulting in a floating point underflow. Again, we apply the same trick as with the odds. We will use the following property of the logarithm to change the product in (3.17) to a sum: $\log(xy) = \log x + \log y$. As shown in Figure 3.7 (b), the logarithm is a monotonic function hence the ranking is not affected by the transformation,

$$RSV_d = \log \prod_{t:d_t=1,q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:d_t=1,q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)} \tag{3.18}$$

The terms in the sum of the above expression are known as the *term relevance weight*. Analogous to the tf-idf term weights, they indicate the discriminating power of a term to separate relevant from non-relevant documents.

In practice, we do not have the necessary information to determine the model parameters p_t and u_t. Since, this would imply that for each query we would have all the information about which documents are relevant to that query and which not. We refer to the set of documents that are relevant to the query as R. The theoretic approach is to learn the parameters by means of relevance feedback from the user (cf. [Crestani et al., 1998; Fuhr, 1992]). Assuming that the system has already ranked the documents with respect to the query (i.e. we have guessed initial estimates for p_t and u_t). The user specifies some documents in the ranking to be relevant and some to be non-relevant. We will refer to this subset as V. V is partitioned into two sets: $VR \subset R$ and $VN \subset N - R$, where N denotes the set of documents in the collection. The simplest way to estimate the parameters p_t and u_t is to use the *maximum likelihood estimate* (i.e. estimating the probability based on the relative frequency). Furthermore, under the assumption

that there exists a very small number of relevant documents compared to the number of non-relevant documents, it seems reasonable to approximate the probability of a term appearing in a non-relevant document u_t by using statistics of the whole collection (cf. [Manning et al., 2008, p.209]). This gives,

$$u_t = \frac{df_t}{N} \tag{3.19}$$

where df_t is the document frequency of term t (i.e. the number of documents in the collection in which t occurs). For the probability of a term appearing in a relevant document p_t we will use the relevance feedback data provided by the user,

$$p_t = \frac{|VR_t|}{|VR|} \tag{3.20}$$

where VR_t is the subset of documents in VR that contain term t. In practice, the maximum likelihood estimate turns out to be too extreme given the usually very small size of VR (some of the query terms might even not occur in VR resulting in a zero probability). Hence, a general technique called *smoothing* is employed which takes some of the probability mass and distributes it according to some prior belief. This results in a decrease of the probability for seen events and an increase for unseen events. In general, a generalization of Laplace smoothing is used, known as Lidstone smoothing, which corresponding to a uniform prior over the unseen events. The resulting *maximum a posterior estimate* is given by,

$$p_t = \frac{|VR_t| + \delta}{|VR| + 2\delta} = \frac{|VR_t| + \frac{1}{2}}{|VR| + 1} \tag{3.21}$$

where $\delta = \frac{1}{2}$ is used to denote our belief in uniformity. Given the new estimate of p_t, the system re-ranks the documents and the new ranking is presented to the user. The process must not stop here, the user might proceed by again judging some of the documents as relevant or not.

In contrast to the Rocchio algorithm the probabilistic approach to relevance feedback presented in this section does only attempt to reweight the query terms. That is, we only estimate the p_t and u_t for terms t in the query (the summation over all terms t where $q_t = 1$ and $d_t = 1$ in (3.18)). We might, however, expand the query with terms from the documents judged as relevant as outlined in Section 2.3.1 before estimating the model parameters p_t and u_t.

The BIM as described above is essentially the same as the well known (Bernoulli) Naive Bayes classifier that is frequently used in text categorization (cf. [Manning et al., 2008, p.204]). The only major difference is that the latter does not employ the odds transformation since it builds upon a different decision theoretic principle (i.e. the Bayes decision-rule).

3.2.2 The Binary Independence Indexing Model

The binary independence indexing model (BII) [Fuhr and Buckley, 1991] derives probabilistic document term weights based on data obtained from relevance feedback provided by the user. The model is a variant of the probabilistic indexing model of Maron and Kuhns [1960], the very first probabilistic model of IR (cf. [Fuhr, 1992]). The BII is an instance of the document-related learning approach hence it regards a single document with respect to a number of queries.

In contrast to the BIM, the BII model represents a query q as a binary term incidence vector $\vec{q} = (q_1, \ldots, q_M)$, where $q_i \in \{0, 1\}$ and M is the vocabulary size. The model does not specify the document representation, similar to query representation in the BIM. However, following

Fuhr [1992], throughout this section we will assume that a set of terms exists for which we will derive weights with respect to each document in the collection. A document d is represented by a term vector $\vec{d}$, each component of the vector refers to term weight w.r.t. d. As the BIM, the BII model aims to rank the documents in the collection in decreasing order of their probability of relevance $P(R = 1|\vec{d}, \vec{q})$, hence, satisfying the probabilistic ranking principle. In order to estimate this probability we first apply Bayes' theorem, given by

$$P(R = 1|\vec{q}, \vec{d}) = \frac{P(\vec{q}|R = 1, \vec{d})P(R = 1|\vec{d})}{P(\vec{q}|\vec{d})} \tag{3.22}$$

Here, $P(R = 1|\vec{d})$ is the probability that a document with representation $\vec{d}$ is judged relevant and $P(\vec{q}|\vec{d})$ is the probability that we see the query representation $\vec{q}$ (i.e. the probability that the query is issued). $P(\vec{q}|R = 1, \vec{d})$ refers to the probability that given a relevant document representation $\vec{d}$ what is the probability that $\vec{q}$ was the query. Notice the similarity between the above formula and (3.11). In the BII model we condition on the document representation $\vec{d}$. Whereas, in the BIM we condition on the query representation $\vec{q}$.

As in the BIM, $P(\vec{q}|R = 1, \vec{d}) = P(q_1, \ldots, q_M|R = 1, \vec{d})$ requires to estimate the complex joint probability distribution of query terms. Again, we make use of the Naive Bayes conditional independence assumption in order to make the parameter estimation computationally tractable. Assuming that query terms are conditionally independent we get,

$$P(\vec{q}|R = 1, \vec{d}) = \prod_{i=1}^{M} P(q_i|R = 1, \vec{d}) \tag{3.23}$$

Substituting (3.23) into (3.22) and again applying Bayes' theorem to the factor in (3.23) gives,

$$P(R = 1|\vec{q}, \vec{d}) = \frac{P(R = 1|\vec{d})}{P(\vec{q}|\vec{d})} \cdot \prod_{i=1}^{M} P(q_i|R = 1, \vec{d}) \tag{3.24}$$

$$= \frac{P(R = 1|\vec{d})}{P(\vec{q}|\vec{d})} \cdot \prod_{i=1}^{M} \frac{P(R = 1|q_i, \vec{d})P(q_i|\vec{d})}{P(R = 1|\vec{d})} \tag{3.25}$$

Here, the probabilities $P(q_i|\vec{d})$ and $P(\vec{q}|\vec{d})$, that is the probability that a specific query term and a specific query are issued, respectively, are constant among all documents with respect to a query. Hence, they can be ignored since they do not alter the ranking of the documents with respect to a query. Thus, removing constant terms and splitting the product in (3.24) into one product for absent and present terms, respectively, we get,

$$P(R = 1|\vec{q}, \vec{d}) \propto P(R = 1|\vec{d}) \cdot \prod_{q_i=1} \frac{P(R = 1|q_i = 1, \vec{d})}{P(R = 1|\vec{d})} \cdot \prod_{q_i=0} \frac{P(R = 1|q_i = 0, \vec{d})}{P(R = 1|\vec{d})} \tag{3.26}$$

The above formula takes into account all terms in the vocabulary. As pointed out in Section 3.2.1 we prefer to solely consider the query terms. Following Maron and Kuhns [1960] we simply assume that the probability that a document representation is judged relevant with respect to a query representation depends solely on the terms present in the query hence $P(R = 1|q_i = 0, \vec{d}) = P(R = 1|\vec{d})$. It follows that the rightmost product in (3.26) amounts to 1 and can be ignored. The numerator of the left product in (3.26), $P(R = 1|q_i = 1, \vec{d})$, is known as the

probabilistic index term weight for term t_i in the document $\vec{d}$. Assuming that $P(R = 1|q_i = 1, \vec{d}) = P(R = 1|d_m)\ \forall t_i \notin d$, that is the probabilistic index term weight of a term not present in the document representation is simply the prior probability of the document representation judged relevant, the final BII formula is given by,

$$P(R = 1|\vec{q}, \vec{d}) \propto P(R = 1|\vec{d}) \cdot \prod_{i:d_i=1,q_i=1} \frac{P(R = 1|t_i, \vec{d})}{P(R = 1|\vec{d})} \tag{3.27}$$

The probabilities $P(R = 1|\vec{d})$ and $P(R = 1|t_i, \vec{d})$ are estimated from relevance feedback data similar to the BIM (e.g. by MLE). The major difference to the BIM is that in the BII we regard a number of queries with respect to a single document in order to derive the necessary parameters as opposed to a number of documents with respect to a single query. However, according to Fuhr [1992] the above formula can hardly be applied in practice due to insufficient training data.

In order to overcome this deficiencies Robertson et al. [1982] proposed a unification of the BIM and the BII model called Model 3 (cf. [Crestani et al., 1998]). However, according to Crestani et al. [1998], "*a computationally tractable estimation theory fully faithful to Model 3 has not been proposed.*". [Fuhr, 1989] and [Wong and Yao, 1989] further explored the ideas proposed by Robertson (cf. [Fuhr, 1992]).

3.2.3 The Darmstadt Indexing Approach

In order to overcome the problems of the BII model the Darmstadt Indexing Approach was proposed [Fuhr, 1989; Fuhr and Buckley, 1991]. The DIA is based on the abstraction of representations of queries, documents and terms to so-called descriptions. This description-oriented indexing approach learns a mapping from descriptions to probabilistic indexing weights. The indexing process is divided into two steps: (1) a description step and (2) a decision step.

In the description step, descriptions for term-document pairs (t_i, d_m) are derived. A (relevance) description $x(t_i, d_m)$ consists of features referring to properties of the term, the document and their relationship (cf. [Fuhr, 1992]). According to Fuhr [1992], one of the key advantages of the DIA is that the definition of the mapping x can be adapted to the specific application context. Hence, the mapping can take into account the representation of documents and the amount of learning data available. Table 3.3 shows the feature mapping that was used in [Fuhr and Buckley, 1991]. A relevance description can be thought of as a feature vector, making an analogy to pattern recognition, describing the term, the document and their relationship.

In the decision step, a probabilistic index term weight for the relevance description is derived. Hence, instead of estimating $P(R = 1|t_i, \vec{d})$ as in the BII model the DIA estimates the probability $P(R = 1|x(t_i, \vec{d}))$. In the BII we have to estimate the probabilistic index term weight of term t_i and document $\vec{d}$ by regarding $\vec{d}$ with respect to all queries containing t_i. In the DIA, however, we abandon the document-related learning approach of the BII and instead employ a description-oriented learning approach. This difference is illustrated in Figure 3.8. For each query-document pair in the learning sample we form relevance descriptions for the terms occurring in both query and document as in the description step. This multi-set of feature vectors along with the corresponding binary relevance judgements $\langle x(t_i, \vec{d}_m), R \in \{0, 1\}\rangle, \ldots$ can be used to directly estimate the probabilistic index term weight $P(R = 1|x(t_i, \vec{d}))$ based on the relative frequency of relevance descriptions. According to Fuhr [1992] better estimates

Relevance description $\vec{x}$	Feature
x_1	$tf_{m,i}$, the within-document frequency of t_i in d_m
x_2	$\frac{1}{\max tf_m}$, the inverse of the maximum within-document frequency in d_m
x_3	$-\log \frac{df_i}{N}$, the inverse document frequency (idf) of t_i.
x_4	$\log \lvert d_m \rvert$, the logarithm of the length of d_m.
x_5	$ta_{m,i} = \begin{cases} 1 & \text{if } t_i \text{ occurs in the title of } d_m \\ 0 & \text{otherwise} \end{cases}$

Table 3.3: Relevance description $\vec{x}$ used in [Fuhr and Buckley, 1991]. The table shows the components of the feature vector $\vec{x}$ referring to properties of term-document pairs.

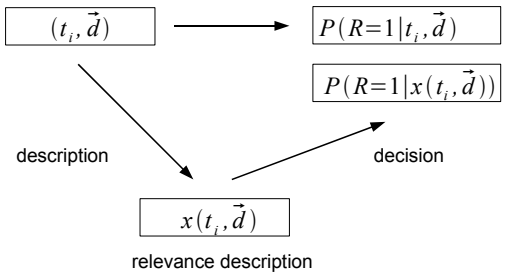

Figure 3.8: Instead of estimating $P(R = 1|t_i, \vec{d})$ directly as in the BII model, the DIA subdivides the indexing task in a description step and a decision step (adapted from [Fuhr, 1992]).

can be achieved by employing probabilistic classification methods from machine learning and pattern recognition. Hence, to approximate $P(R = 1|x(t_i, d_m))$ using some indexing function $e(x(t_i, d_m))$. The indexing function can be applied to terms and documents not seen in the learning sample. Hence, the DIA makes the best possible use of the relevance feedback data available. According to [Fuhr, 1992], "*the major advantage of this indexing approach is its flexibility w.r.t. the representation of documents, which becomes important when advanced text analysis methods are used (e.g. noun phrases in addition to words, see for example [Fuhr, 1989])*".

In earlier work, Fuhr [1989] uses least-squares polynomial regression (LSP) [Bishop, 2006, p.137] as the indexing function. However, Cooper et al. [1992] criticized using polynomial regression to estimate the probability of relevance. Rather, Cooper pointed out that *logistic regression* is a much more appropriate tool when the dependent variable (the random variable R) is dichotomous, either relevant (1) or not (0). Fuhr and Buckley [1991] reports on the usage of several probabilistic classification algorithms as indexing functions. Fuhr and Pfeifer [1994] experimentally compared logistic regression and LSP regression and found that they yield identical results.

Chapter 4

Neural Network Retrieval Models

4.1 Neural Networks

Neural networks (NN) are mathematical models intended to resemble our belief of information processing in biological systems (i.e. the human brain). A neural network is essentially a graph, nodes represent *neurons* and edges represent *synaptic connections* between neurons. Neural networks have been studied extensively in the area of subsymbolic artificial intelligence where they have been used for both, supervised (e.g. multilayer perceptrons) and unsupervised learning (e.g. self-organizing maps, neural gas). Nowadays, the link to the brain is not that important anymore, rather they are analyzed from a viewpoint of statistics.

In general, we distinguish between two fundamental types of neural networks: (a) *recurrent* and (b) *feed-forward* networks. Recurrent neural networks impose no restrictions on the structure of the graph. In IR recurrent neural networks (e.g. Hopfield networks [Mackay, 2002, p.505ff]) have been used in the late 1980s by various groups ([Belew, 1989; Wilkinson and Hingston, 1991]). Most of the used networks had the simple structure of a bi-partite graph [Wilkinson and Hingston, 1991]. Such networks are used for (noisy) pattern matching and referred to as *bi-directional associative memories* [Kosko, 1988]. They are briefly discussed in Section 4.2.1.

The remaining section presents feed-forward neural networks, the most widely used type of NN for pattern recognition, and network training using the *error backpropagation algorithm*, a generalization of the well known *delta-rule* for multi-layer networks. Feed-forward neural networks have been used in a number of retrieval systems [Mandl, 2000; Crestani and van Rijsbergen, 1997; Schütze et al., 1995; Crestani, 1993] and form the basis for the connectionist model presented in this thesis.

4.1.1 Feed-Forward Neural Networks

A *feed-forward neural network* is a special case of neural network which underlying graph is directed and acyclic (i.e. a DAG). They are the most successful (i.e. practical) type of neural networks and have been used extensively in the area of pattern recognition. Feed-forward neural networks are also known as *multi-layer perceptrons* although this is not an accurate denomination, since they actually implement multiple layers of *logistic regression* models rather than multiple layers of perceptrons (cf. [Bishop, 2006]).

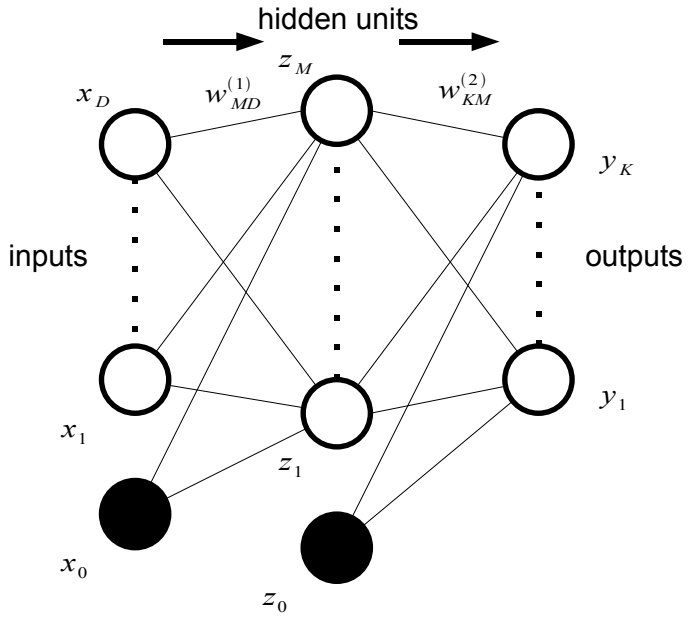

Figure 4.1: Network diagram of a two-layer (two weight layers $w^{(1)}_{ji}$ and $w^{(2)}_{kj}$) feed-forward neural network. Information propagates from the input to the output layer (as denoted by the arrows). Adapted from Bishop [2006].

A feed-forward network consists of multiple layers of nodes $S = 1, \ldots, N$, where the nodes of layer i are fully connected to the nodes of layer $i+1$ for $i = 1, \ldots, N-1$. The layers 1 and N are referred as the input and output layers, respectively. All layers in between are called *hidden layers*. The network diagram in Figure 4.1 depicts a feed-forward network with three node layers (i.e. input, hidden and output layer). However, in general such a network is denoted as a two-layer feed-forward network since there are two layers of *adaptive parameters* (weights), $w^{(k)}_{ji}$, where the superscript k denotes the corresponding weight layer. Subsequently, except explicitly stated, an n-layered network refers to the number of layers of adaptive parameters of the network. Two-layered networks are the most common type of feed-forward neural networks, networks with more than three-layers (i.e. more than two hidden layers) are not commonly used. The subsequent formal definition of feed-forward networks follows Bishop [2006], since it represents one of the most authoritative and up-to-date sources on NN for pattern recognition.

A feed-forward neural network model can be described as a series of functional transformations. First M linear combinations of the input variables $x_1, \ldots, x_D$ are constructed according to

$$a_j = \sum_{i=1}^{D} w^{(1)}_{ji} x_i + w^{(1)}_{j0} \tag{4.1}$$

where $j = 1, \ldots, M$ and M is the number of *hidden units*, that is the number of nodes in the hidden layer. The variables $w^{(1)}_{ji}$ refer to the weights in the first layer (i.e. the first layer of adaptive parameters). The parameters $w^{(k)}_{j0}$ are called *biases*. The bias terms allow for any fixed offset in the data (cf. [Bishop, 2006]) and can be thought of as a combination with an additional dummy parameter that is equal to 1. The dummy parameters are depicted in Figure 4.1 by the filled nodes.

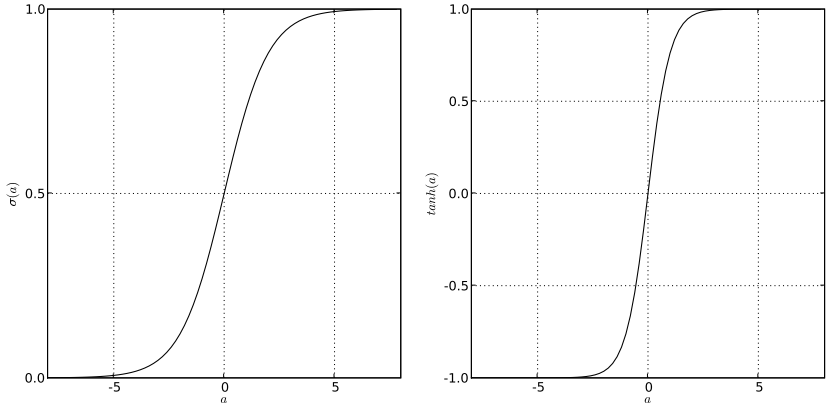

Figure 4.2: Plots of (a) the logistic sigmoid function $\sigma(a) = \frac{1}{1+e^{-a}}$ and (b) the tangents hyperbolicus $\tanh(a) = 1 - \frac{2}{1+e^{2a}}$

Each of those linear combinations is then transformed using a differentiable nonlinear *activation function* $h(\cdot)$ resulting in

$$z_j = h(a_j) \tag{4.2}$$

These quantities are referred to as the outputs of the hidden units. Among the most frequently used activation functions for the hidden units are the logistic sigmoid (*logsig*) and the tangents hyperbolicus (*tanh*). Figure 4.2 shows both of them. In the second transformation the hidden units are again linearly combined in order to get the activation of the *output units*

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{4.3}$$

where $k = 1, \ldots, K$ and K is the total number of outputs. In the final transformation the activations of the output units are transformed using an activation function which is appropriate for the task the network is intended to perform. For regression (i.e. mapping to the reals), the natural choice for the output activation function is the identity so that $y_k = a_k$. For multiple binary classification problems, the logistic sigmoid function is used to transform the activation of the output units so that

$$y_k = \sigma(a_k) \tag{4.4}$$

where

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{4.5}$$

Since the logistic sigmoid function given by Equation (4.5) is constrained to the interval $(0, 1)$ the output can be interpreted as a probability estimate $P(t = 1|\mathbf{x}, \mathbf{w})$. That is, the conditional probability of $t = 1$ (i.e. the positive class) given the input vector $\mathbf{x}$ and the model $\mathbf{w}$. Hence, the function can be regarded as a membership function returning the *probability of class membership* of the input pattern x (cf. [Witten and Frank, 2005]).

By combining the various transformations and assuming that the output activation function

is the logistic sigmoid function the output of the overall network function is given by

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right) \tag{4.6}$$

where the weights and the biases have been grouped together into a weight vector $\mathbf{w}$. The vector $\mathbf{x}$ is often called the input *pattern* or *instance*, its components are the input parameters $x_1, \ldots, x_D$, where D is the dimensionality of the input space.

The evaluation of Equation (4.6) can be regarded as a *forward propagation* of information through the network, starting from the input layer propagating to the output layer as illustrated by the two arrows in Figure 4.1. As can be seen by the form of Equation (4.6) a feed-forward neural network is simply a nonlinear function mapping from an D dimensional input space to a K dimensional output space. According to Bishop [2006], it can be shown, that "*a two-layer network with linear outputs can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy provided the network has a sufficiently large number of hidden units.* "

4.1.2 Error Backpropagation

The process of determining the network parameters is called *network training*. In case of supervised learning we are presented a *training set* comprising a number of input vectors $\mathbf{x_1}, \ldots, \mathbf{x_N}$ along with their corresponding target vectors $\mathbf{t_1}, \ldots, \mathbf{t_N}$ which have been generated by some unknown function f. The goal of network training is to approximate this function with a function $\hat{f}$ (the *hypothesis*) based on the available training data. In this sense, the process of network training can be regarded as a search problem: Given a parameter space we are looking for a concrete parameter setting (a weight vector $\mathbf{w}$) which minimize a given *error function* (or sometimes called *loss function*). The error function serves as a proxy for the goodness of approximation.

The choice for the error function is subject to the task at hand. For regression the *sum-of-squares error function* is used, given by

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} ||\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n||^2 \tag{4.7}$$

For binary classification one usually chooses the *cross-entropy error function* which is defined as the negative log-likelihood of the data,

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{\mathbf{t_n} \ln \mathbf{y_n} + (1 - \mathbf{t_n}) \ln(1 - \mathbf{y_n})\} \tag{4.8}$$

where $\mathbf{y_n} = y(\mathbf{x}_n, \mathbf{w})$ is used to denote the prediction of the network.

An example of a supervised learning problem is depicted in Figure 4.3. It shows a number of inputs $\mathbf{x_i}$ along with their corresponding target values $\mathbf{t_i}$. The input data was generated by choosing values spaced uniformly in range $[0, 1]$. The target values $\mathbf{t_i}$ were obtained by evaluating the function $\sin(2\pi x)$ and then adding uniform gaussion noise (zero mean, 0.2 variance). The function $\sin(2\pi x)$, shown in light grey, is the function we want to approximate given only the training data points. The black curve shows the approximation of a feed-forward neural

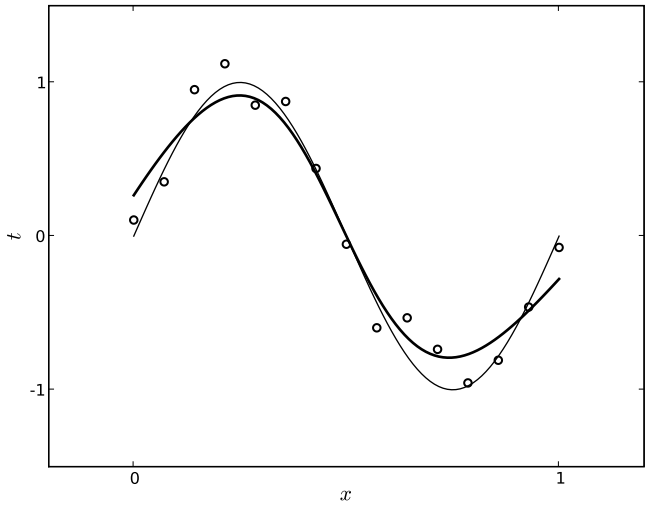

Figure 4.3: Illustration of the function approximation capability of a two-layer feed-forward neural network with 10 hidden sigmoidal units (thick curve). The input data was generated by choosing values of x spaced uniformly in range $[0, 1]$. The target values were generated by the function $y = \sin(2\pi x)$ (the thin curve) with some uniform gaussian noise added.

network with 10 sigmoidal hidden units trained on the 15 training examples using the sum-of-squares error function (4.7).

The general approach to network training is to employ an iterative procedure based on gradient information to update the weight vector $\mathbf{w}$. At each iteration of the *gradient descent* procedure the derivative of the error function with respect to the network parameters $\mathbf{w}$ is evaluated and the weights are updated to comprise a small step in the direction of the negative gradient. That is, "*the direction of the greatest rate of decrease in the error function*" [Bishop, 2006, p.240].

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla E(\mathbf{w}^{(t)}) \tag{4.9}$$

where $\eta > 0$ is a parameter known as the *learning rate* that controls the step size in the parameter space. The above algorithm will eventually reach a minimum of the error function. However, there is no assurance if the obtained solution represents the global minimum, but some local minimum of the error function. Note that in the above formula the gradient is evaluated with respect to the whole training set (the summation in (4.7) and (4.8), respectively). Such a setting is called *batch learning* because the whole training set needs to be processed. There is also an *online* version of gradient descent, known as sequential or *stochastic gradient descent*. Stochastic gradient descent updates the weight vector based on one training example at a time,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla E_n(\mathbf{w}^{(t)}) \tag{4.10}$$

where E_n is the summand of (4.7) and (4.8), respectively, corresponding to input pattern n. The update is performed for each example in the training set. Online gradient descent has proven useful in practice for training neural networks on large datasets [Bishop, 2006, p.240]. Furthermore, online gradient descent is rather unstable compared to its batch version, leading to

the possibility to escape from local minima of the error function. According to Bishop [2006, p.240], gradient decent is a poor choice for batch optimization. There are more efficient methods, such as *conjugate gradients* and *quasi-Newton* methods[1].

Each iteration of the training procedure comprises two steps: (1) the derivative of the error function with respect to the weight vector $\mathbf{w}$ is evaluated and (2) the derivative is used to compute the adjustment of the weight vector. The last step is a straight forward evaluation of (4.9). In contrast, the first step seems rather demanding. However, it turns out that there is a computationally efficient method for evaluating such derivatives, known as *error backpropagation* (or simply backprop). The remaining of this section is dedicated to the derivation of the backpropagation algorithm for an arbitrary feed-forward neural network with arbitrary (differentiable) activation functions and error functions. Subsequently, only the problem of evaluating $\nabla E_n(\mathbf{w})$ is considered. For the batch setting one simply has to accumulate the error over all training examples,

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}) \tag{4.11}$$

The backpropagation technique starts with a forward propagation of the input vector for a given training example as described in detail in the above section. In the forward propagation process, each unit j in the network computes a_j a weighted sum of its inputs according to (4.3). Subsequently, a_j is transformed by a nonlinear activation function to give the activation z_j of the unit according to (4.2). These quantities are stored because they are needed to evaluate the derivative. The derivative of each weight w_{ji} individually is evaluated by applying the chain rule for partial derivatives,

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{4.12}$$

The second partial derivative is simply the derivative of (4.3) with respect to w_{ji}, given by

$$\frac{\partial a_j}{\partial w_{ji}} = \frac{\partial w_{ji} z_i}{\partial w_{ji}} = z_i \tag{4.13}$$

For the first partial derivative usually the following notation is introduced,

$$\frac{\partial E_n}{\partial a_j} \equiv \delta_j \tag{4.14}$$

Substituting (4.14) and (4.13) into (4.12) gives,

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{4.15}$$

So the derivative with respect to a connection weight w_{ji} is simply the product of the activation of the source node z_i and the value of δ of the target node.

The δ values of the output units are subject to the activation function $h(\cdot)$ and the error function used. It turns out that there is a natural choice for the activation function of the output units given either (4.7) or (4.8) as the desired error function. For regression, we choose the sum-of-squares error function leading to linear output units. For multiple independent binary

[1]The neural network used to approximate the function in Figure 4.3 was trained using a conjugate gradients method provided by [Jones et al., 2001].

classification decisions we choose the cross-entropy error function hence logistic sigmoidal output units. Both choices give the same value of δ for the output units $1 \leq k \leq K$,

$$\delta_k = y_k - t_k \tag{4.16}$$

Hence, for the output units the value of δ is simply the *error* (the difference between the predicted and the actual value). For a hidden unit j, the value of δ_j is obtained by propagating back the errors δ_k of the units higher up in the network hierarchy,

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{4.17}$$

By recursively applying the above formula the value of δ for all hidden units can be obtained regardless of the topology of the feed-forward network [Bishop, 2006]. A pseudocode implementation of the backpropagation algorithm is shown in Algorithm 1.

Algorithm 1 Network training using the error-backpropagation algorithm (adapted from [Russell and Norvig, 2002]).

input *examples*, a set of examples, each with input vector x and target vector t
input *network*, a multi-layer network with M layers, weights w_{ji} and activation function h.
  function BACKPROP-LEARNING(examples, network)
    repeat
      for all e in *examples* **do**
        for all node j in input layer **do**
          $z_j \leftarrow x_j[e]$
        for $l = 2$ **to** M **do** ▷ Forward propagation
          $\mathrm{a}_j \leftarrow \sum_i w_{ji} z_i$
          $z_j \leftarrow h(a_j)$
        for all node k in output layer **do**
          $\delta_k \leftarrow z_k - t_k[e]$
        for $l = M - 1$ **to** 1 **do** ▷ Error backpropagation
          for all node i in layer l **do**
            $\delta_i \leftarrow h'(a_i) \sum_j w_{ji} \delta_j$
            for all node j in layer $l + 1$ **do**
              $w_{ji} \leftarrow w_{ji} - \eta \delta_j z_i$ ▷ Update weights
    until some stopping criterion is met
    return NEURAL-NET-HYPOTHESIS

As stated above the major advantage of the backpropagation algorithm is efficiency. The runtime complexity of the algorithm is $O(W)$ where W is the number of adaptive parameters in the network. However, this refers to a single evaluation of $E_n(\mathbf{w})$ for a given input. Network training using gradient descent requires the backpropagation procedure to be invoked N times for each of M iterations. Where N refers to the number of input patterns $\mathbf{x}$ and M is the number of iterations the gradient descent procedure needs to converge, known as the *epochs*. In practice, convergence is not required rather a fixed number of iterations is performed. Often, a method known as *early stopping* is employed that stops the iterative process as soon as the error on some hold-out data is getting worse. This method aims to maintain the generalization capabilities of the network hence to avoid overfitting.

Since W is usually much larger than either N or M network training based on gradient descent is still linear in the number of parameters. However, in practice the constants N and M are rather large resulting in a long training time. Again, it has to be noted that the error backpropagation algorithm is simply a method to evaluate the derivative of the error function with respect to the weights. Hence, it can be used in any optimization procedure that is based on gradient information, not just gradient descent. As stated above, in the case of batch learning one usually employs other optimization techniques requiring much less evaluations of the derivative than gradient descent. For more information on advanced topics in network training such as *regularization* and more efficient optimization techniques see Bishop [2006].

4.2 Retrieval Models based on Neural Networks

In the vector-space model of information retrieval, document and query representations are matched in order to compute a ranking. The use of neural networks for IR seems consequential, since they are well known for their pattern matching capabilities. However, NN hold another property that is beneficial to the task of IR: connectionist models excel in the presence of noisy data. According to Biron and Kraft [1995], the utility of connectionist models in information retrieval stems from the fact that the uncertainty in document indexing, query formulation and relevance judgements can be regarded as a form of noise.

Since the renaissance of neural networks (i.e. connectionism) in the mid 1980s, there have been numerous attempts to create retrieval models based on neural networks [Belew, 1989; Kwok, 1995; Wilkinson and Hingston, 1991; Crouch et al., 1994]. Most attempts can be grouped into three major classes:

- Spreading Activation Models
- Connectionist Models
- Transformation Networks

Spreading activation models are by far the most common approach (cf. [Baeza-Yates and Ribeiro-Neto, 1999, p.46]), however, in fact they are not proper neural networks. Crestani [1997] pointed out two aspects that distinguish neural networks from spreading activation models: First, SA models lack an explicit learning procedure which adapts the structure of the network to reflect some desired activation pattern. Adaptivity is one of the key aspects of connectionist models. Second, the presence of non-linear activation functions. Although there have been attempts to employ non-linear activation functions in SA models, in general, they use the identity function.

Many people in the IR community use the term spreading activation models to denote all of the above models since the ”spreading of activation” is their least common denominator. However, following Mandl [2000], in this thesis the term spreading activation model refers to the pure form as described in the next section. The remainder of this chapter is dedicated to an in-depth discussion of the three major classes.

4.2.1 Spreading Activation Models

As stated above spreading activation models are basically neural networks lacking an explicit learning procedure. Most SA approaches make use of a layered architecture where the nodes

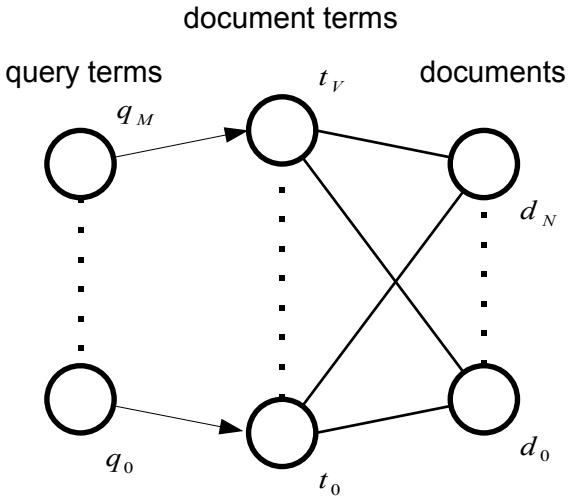

Figure 4.4: The SA model proposed by Wilkinson and Hingston [1991]. The right part (document term and document layer) resembles a bi-directional associative memory in which activation propagates "back-and-forth".

represent information items (e.g. queries,terms, documents) and the layers represent homogeneous groups of those items [Wilkinson and Hingston, 1991; Crouch et al., 1994]. Unlike feed-forward networks, spreading activation models in general employ bi-directional connections. They share properties with so called *bi-directional associative memories* [Kosko, 1988]. That are recurrent neural networks comprising two node layers that recurrently propagate signals between each other. The connection weights between input and output layer are usually symmetric.

Figure 4.4 shows the structure of the spreading activation model proposed by Wilkinson and Hingston [1991]. The depicted model is characteristic for most spreading activation models. The network consists of an input layer which contains one node for each query term. The sole purpose of the first weight layer is to associate query terms with index terms. In general, there is a one to one correspondence between query and index terms. Hence, the query layer can be ignored for most retrieval scenarios (cf. [Mandl, 2000]). The right part of the network associates the index terms with the documents. It resembles an heterogeneous bi-directional associative memory. Activation spreads from the index term nodes to the document layer and activates the document nodes. At the next clock tick the activation of the document nodes spreads back to the index term layer possibly activating terms which have not been activated initially. In the process documents might be activated that are not indexed with the query terms. This can be thought of as a form of local query expansion (see Section 2.3.1). This property makes the model especially appealing. The process continues for a predefined number of cycles or until a stable state is reached. The ranking of the documents is given by the final activation level of the document nodes in the output layer.

The model intuitively supports relevance feedback by enabling the user to assess and change the activation level of document nodes. The user might increase the activation level of documents identified as relevant and decrease the activation level of documents identified as non-relevant. However, this type of feedback signal only influences the activation pattern and can

be regarded as a form of *transient* learning (i.e. for one query session) as opposed to the general intuition of learning in neural networks which is *persistent* (i.e. changing the connection weights).

Up to now nothing was stated on how to obtain the connection weights if we do not explicitly learn them from training data. The answer is rather simple: traditional IR heuristics as described in Section 3.1.1. Hence, spreading activation models can be regarded as heuristic models similar to the vector-space model. Indeed, many researchers have looked at the similarity between the SA and the VSM model [Mothe, 1994]. If we assume that the initial activation level of each query term k is equal to 1 and fix the connection weights $w_{i,q}^{(1)}$ of the index term i associated with the query term k to be

$$w_{i,k}^{(1)} = \begin{cases} \frac{\text{tf-idf}_{i,q}}{\sqrt{\sum_{t=1}^{V} \text{tf-idf}_{t,q}^2}}, \text{if } i = k \\ 0, \text{otherwise} \end{cases} \tag{4.18}$$

where V is the vocabulary size (i.e. the number of index terms) and $\text{tf-idf}_{i,q}$ is the weight of term i in the query q according to the tf-idf weighting scheme (3.4).

When the signal propagates from a query term node k to the index term node i it gets damped by the connection weight $w_{i,k}^{(1)}$. Assuming that the activation function of the index term nodes is the identity function ($y = x$) and fixing the connection weights $w_{j,i}^{(2)}$ of index term node i and document node j to be

$$w_{j,i}^{(2)} = \frac{\text{tf-idf}_{i,j}}{\sqrt{\sum_{t=1}^{V} \text{tf-idf}_{t,j}^2}} \tag{4.19}$$

where $\text{tf-idf}_{i,j}$ is the weight of the index term i in the document j according to the tf-idf weighting scheme (3.4).

Again the signal gets damped by the connection weight $w_{j,i}^{(2)}$ while propagating to the document nodes. The total activation level of document node d_j is the sum of all signals that reach d_j and given by

$$\sum_{i=1}^{V} w_{i,k}^{(1)} w_{j,i}^{(2)} = \frac{\sum_{i=1}^{V} \text{tf-idf}_{i,q} \text{ tf-idf}_{i,j}}{\sqrt{\sum_{i=1}^{V} \text{tf-idf}_{i,q}^2} \sqrt{\sum_{i=1}^{V} \text{tf-idf}_{i,j}^2}} = \frac{\vec{q} \cdot \vec{d_j}}{\|\vec{q}\| \, \|\vec{d_j}\|} \tag{4.20}$$

where $\vec{q}$ and $\vec{d_j}$ are Vdimensional vector representations of the query and the document j with components equal to $\text{tf-idf}_{i,q}$ and $\text{tf-idf}_{i,j}$, respectively.

Thus, the activation level after the first forward propagation is essentially the cosine similarity as described in (3.2) (cf. [Baeza-Yates and Ribeiro-Neto, 1999, p.47]). In a sense, the model outlined above can be considered as a parallel implementation of the vector-space model (cf. [Biron and Kraft, 1995]). However, the process must not stop here, as stated above, the spreading activation process might continue propagating activation back to the index term nodes. Hence, modifying the initial ranking analogous to the pseudo relevance feedback technique mentioned in Section 2.3.3. Wilkinson and Hingston [1991] report that it is even more effective to inhibit document nodes with an activation level below some threshold to propagate activation back. Boughanem et al. [1999] describe a similar technique - they call it relevance back-propagation - which has been successfully applied in numerous TREC participations of the group [Boughanem and Dupuy, 1996; Boughanem et al., 2003].

Several extensions to the general SA model have been proposed. Boughanem and Dupuy [1996] incorporate connections between index terms themselves. The weights of the inter-term connections might be based on term co-occurrence [Boughanem and Dupuy, 1996] or domain knowledge (e.g. from a domain ontology or a thesaurus) [Scheir, 2008]. Inter-layer connections, however, yield a highly recurrent network. Hence, making the spreading activation process much more complicated (cf. [Crestani, 1997]). In order to ease the situation many researchers impose constraints on the pattern of activation. Typical constraints are distance constraints causing the signal to fade when it reaches nodes that are many edge traversals away from the node initially activated or fan-out constraints which inhibit the activation of high out-degree nodes to avoid over-spreading. Crouch et al. [1994] proposed a SA model incorporating a hierarchy of index terms. The first term layer represents the root of the hierarchy and is followed by layers representing the intermediate levels of the hierarchy. The term layer representing the leaves of the hierarchy is followed by the document layer.

The major drawback of most of the proposed SA models is the mere fact that they have not been applied to large test collections [Wilkinson and Hingston, 1991; Crouch et al., 1994]. To the author's knowledge only one system based on a pure SA model, MERCURE, has participated at the annual TREC. MERCURE participated regularly at TREC from 1996 to 2003 [Boughanem and Dupuy, 1996; Boughanem et al., 2003]. Throughout the seven years the model was only subject to minor changes (the inter-term connections where abandoned after Boughanem and Dupuy [1996]), only the heuristics for the connection weighting were constantly improved, reflecting major successes in IR such as the BM25 weighting scheme [Robertson et al., 2000].

4.2.2 Connectionist Models

Spreading activation models use static connection weights and regard relevance feedback, if any, as a form of transient learning. While, according to Biron and Kraft [1995],

> *in a connectionist approach to IR relevance judgements can be used as a learning signal for adaptively modifying document indexing, in effect using knowledge of the user's past browsing behavior to influence future retrieval.*

In other words, connectionist approaches to IR use feedback to modify (i.e. adapt) the model in a way to better reflect the user's intuition of relevance. This process of adaption to the user's needs and context is what makes connectionist models appealing to the task of IR. One can think of the adaption as a form of *document-space modification* [Salton, 1989; Brauen, 1969] (i.e. changing the document representation) in order to ensure that the indexing weights better reflect the user's intuition of what makes the document relevant. Hence, in some sense connectionist retrieval models resemble the kind of document focused learning employed by the retrieval with probabilistic indexing models discussed in Section 3.2.2.

Another appealing aspect of this "adaptive indexing" approach has been pointed out by Cunningham et al. [1997],

> *if the modifications are retained, then the averaging of feedback across numerous queries and users constructs a representation of the communal meaning attached to keywords by the users. [...] the problem lies in gathering a large enough amount of feedback from a significant number of users, so that a single idiosyncratic opinion or an eccentric user does not skew the network's output.*

In contrast to the vector space model, there is nothing such as ”the connectionist model” of IR[2]. Rather, various models have been proposed, some of them are aimed at specific retrieval tasks and settings. As a consequence the models (i.e. the topology of the neural network) encode much domain knowledge. Due to the diversity of connectionist approaches to IR this section outlines some of the most influential contributions. Eventually enabling the reader to envision the capabilities and expressiveness of such models.

Adaptive Information Retrieval (AIR)

Belew [1989] was among the first who proposed a connectionist approach to the task of information retrieval. He envisioned and eventually implemented a system, AIR, which uses relevance feedback from its users to change its representation of index terms and documents in order to improve at its task over time. AIR uses a modified Hebbian learning rule to change its representation. Belew's connectionist system learns from the feedback of many users and as a result builds up a *consensual* knowledge representation. The basic representation of the system is a layered recurrent neural network. The nodes in the network represent three types of information items: keywords, documents and authors. Each document shares two directed connections with each author and keyword, respectively. The connection weights are bootstrapped using traditional IR statistics to ensure that the un-trained network has desirable retrieval performance. In detail, the system uses an *inverse frequency* weighting by ensuring that the sum of the weights on all outgoing links is constant. Hence, nodes with many out-links propagate less activation per link than nodes with little out-links.

In addition to connectionist learning, AIR supports relevance feedback as transient learning. Belew states that the primary goal of this is to immediately respond to the user action. Hence, the user gets immediate benefit when providing feedback. Furthermore, Belew [1989] provided preliminary evidence that it is possible to learn semantic relationships such as synonyms and common word stems based on relevance feedback.

Probabilistic Indexing and Retrieval Component System (PIRCS)

Among all connectionist approaches to IR Kwok [1989] takes a special position for various reason. In contrast to most neural network approaches, Kwok [1995] builds upon a sound theory of IR, namely, the theory of probabilistic indexing and retrieval [Maron and Kuhns, 1960; Fuhr, 1989] satisfying the Probabilistic Ranking Principle (see Section 3.2). PIRCS is based on a two-layered feed-forward network comprising a query, term and document layer. Actually, the model implements two feed-forward networks, one from the query node layer to the document node layer and vice versa. Connection weights are asymmetric. As in other models of IR, PIRCS treats queries and documents as objects of the same type, both are represented through index terms. This enables the system to be used in both settings, ad-hoc and routing.

The system employs various learning paradigms and heuristics to determine the network parameters: (1) the principle of document self-recovery, (2) term frequencies, and (3) learning through relevance feedback. The connection weights are initially defined using term statistics and the principle of document self-recovery. First, connection weights are initialized based on term frequency and inverse collection frequency. Subsequently, the principle of document self-recovery is applied. The principle is based on the the intuition that the representation of a document is relevant to itself. Thus, when a document representation is used as a query that document should be relevant. This learning paradigm is unsupervised, the employed learning

[2]As well as there is nothing such as ”the probabilistic model” of IR.

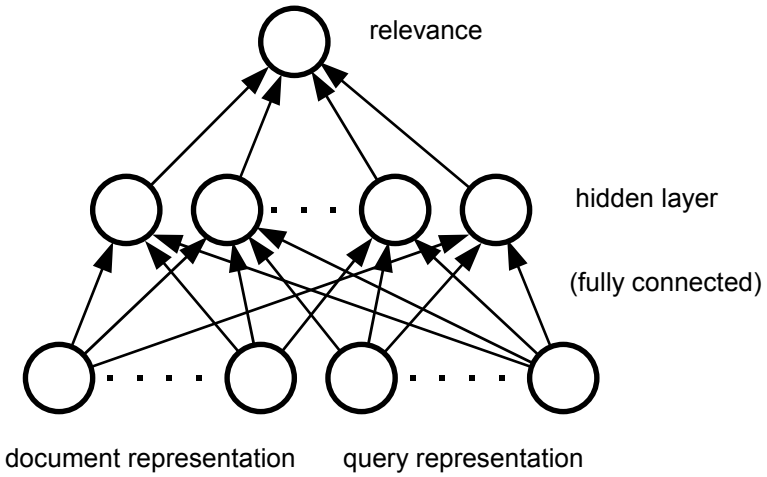

Figure 4.5: The COSIMIR model [Mandl, 2001] implementing the matching between a document and a query representation.

rule is similar to competitive learning with decay [Rumelhart and Mcclelland, 1986]. According to Kwok [1995], a network trained using document self-recovery is comparable in terms of effectiveness to some of the best initial term weighting schemes. In operational mode, the network is trained using relevance feedback.

Besides MERCURE, PIRCS was among the few systems based on NN that regularly participated in large-scale system evaluations at TREC. Throughout the years, PRICS achieved very good results. In TREC-5 it was among the top eight systems for both tasks, ad-hoc and routing (cf. [Voorhees and Harman, 1997]) and was the leading system in the chinese-track. In Trec-6 both PIRCS and MERCURE again were among the top eight systems in the ad-hoc and routing tasks.

Cognitive Similarity Learning in Information Retrieval (COSIMIR)

The COSIMIR model implements the matching between a query representation and a document representation using a two-layer feed-forward network that is trained using error-backpropagation. Basically, the model approximates a similarity function given query-document pairs with their corresponding binary relevance judgements. The similarity of both representations is modeled in a single output unit. The topology of the network is shown in Figure 4.5. Thus, the COSIMIR model can be regarded as an alternative to the widely adopted cosine-similarity measure (3.1).

The input layer of the network comprises one input unit for each query and document index term, respectively. The actual matching is encoded in the hidden layer. In a full text setting the size of the input layer would be twice the size of the vocabulary. For any collection of reasonable size this results in too much parameters to estimate. Consider for example a tiny collection with a vocabulary of 1000 terms and lets assume that 20 hidden units are used, given such a collection the number of parameters in the model amounts to 40,020. According to Russell and Norvig [2002] a reasonable rule of thumb to estimate the number of training examples needed to train the network is 10 examples per parameter. Hence, for our toy example we would need more

than 400,000 query-document pairs with corresponding relevance judgements to ensure that the network is properly trained. Hence, for full text search dimensionality reduction techniques such as latent semantic indexing (LSI) [Deerwester et al., 1990] are vital.

According to Mandl [2000] the system is able to create a similarity function for each user personally. However, due to the lack of sufficient training data, all judgements are used to train a single similarity function for the whole user population.

4.2.3 Tranformation Networks

Transformation network have been introduced by Crestani and van Rijsbergen [1997] as a model for adaptive retrieval. A transformation network is actually not a proper retrieval model, but rather, it is a sub-symbolic knowledge representation which is used to transform a query from the query term space into the document term space. The representation of the query in the document space in subsequently used to query a conventional IR system.

Transformation networks are implemented as a two-layer feed-forward network where the number of input units corresponds to the number of possible query terms and the number of output units corresponds to the number of index terms. For network training the backpropagation algorithm is used. For each input pattern (i.e. for each query) the network is presented a number of output patterns (i.e. relevant documents). A major problem of this approach, which they called *total learning*, is that for a query with k relevant documents, the network is trained on one input pattern with k output patterns. Hence, if k is too small the network might be faced with contradictory patterns. Moreover, due to such contradictory examples the network might learn output patterns that do not exist in practice resulting in poor retrieval quality. In order to tackle this problem Crestani and van Rijsbergen [1997] experimented with two alternative learning methods. The first method, which they called *horizontal learning* trains the network based on the cluster centroid of the relevant documents w.r.t. a query instead of k query-document pairs. The second method, vertical learning, is similar to total learning. However, instead of training the network with all available relevant documents, vertical learning chooses a fixed sized subset of the relevant documents (e.g. 1/3 of the total number of relevant documents w.r.t a query) for learning. They found that the latter method, vertical learning, is superior to both horizontal and total learning.

Crestani and van Rijsbergen [1997] evaluated their approach on a subset of the Cranfield test collection (see Section 2.2). The subset comprises 200 documents and 42 queries with exhaustive relevance judgements. The authors argued that the considerable small size of the test collection was due to the scalability problems of their neural network simulation software. The queries contained a total number of 195 distinct query terms resulting in as many input units. The number of (controlled) index terms in the collection is 1142 resulting in as many output units. The hidden layer comprises 100 hidden units resulting in a total number of more than 130,000 adaptive parameters. Given the sheer size of such a network, the number of training examples seems ridiculously small: 42 in the case of horizontal learning and 280 in the case of vertical learning when we assume that, as an upper bound, all documents in the collection are relevant to all queries. Hence, even in the case of vertical learning the network is heavily under-trained. It follows, that the conclusions drawn by the authors should be taken with a grain of salt. As stated above, the network was trained using the backpropagation algorithm using a constant number of 300 epochs.

In a subsequent experiment the authors used the transformation network for query expansion

by expanding the original query with the highest activated index term nodes. They compared their method, which they called *neural relevance feedback*, to probabilistic relevance feedback (PRF) which refers to the binary independence model presented in Section 3.2.1. In order to select expansion terms for probabilistic relevance feedback they employed a method similar to the Robertson Selection Value (see Section 2.3). According to their findings, "*the performance of PRF is better at any level of training and especially at lower levels*" [Crestani and van Rijsbergen, 1997].

The authors concluded that, besides the acquisition bottleneck of training data, one of the major drawbacks in adopting NN to model IR is the high computational complexity due to the large number of index terms involved to model the matching between the query and documents. Despite of these facts transformation networks have been subject to further research [Mandl, 2000; Cortez et al., 1995].

Chapter 5

A Connectionist Model for Adaptive Information Retrieval

5.1 Associative Information Retrieval

The connectionist retrieval model which has been developed in the course of this thesis builds upon the associative retrieval model introduced by Scheir [2008]. The goal of Scheir was to create an effective model for information retrieval on the Semantic Desktop in the presence of *sparse semantic annotations*. According to [Scheir et al., 2007a],

> *The sparse annotation of resources with semantic information presents a major obstacle in realizing search applications for the Semantic Web or the Semantic Desktop*.

The term Semantic Desktop denotes the incubation of Semantic Web technology on the desktop environment. A throughout discussion of the Semantic Web initiative and related technology is beyond the scope of this thesis, interested readers are encouraged to consult Wahlster et al. [2002] for detailed information on the topics.

In order to overcome the sparse annotation problem, Scheir [2008] employs techniques from associative information retrieval. According to Crestani [1997], associative IR refers to the idea that,

> *it is possible to retrieve relevant information by retrieving information that is "associated" with some information the user already retrieved and that is know it to be relevant*.

In associative retrieval, the associations among information items are usually modelled using weighted directed graphs so-called *associative networks*. The network weights express the degree of association between two items. Retrieval is usually done by means of spreading activation where the activation originates from some items that are known to be relevant possibly activating other items which are "associated" with these items.

The retrieval model, proposed by Scheir [2008], builds upon this associative retrieval paradigm and uses spreading activation (see Section 4.2.1) to retrieve documents given a query comprising a set of concepts from the controlled vocabulary of a domain ontology. From a conceptual

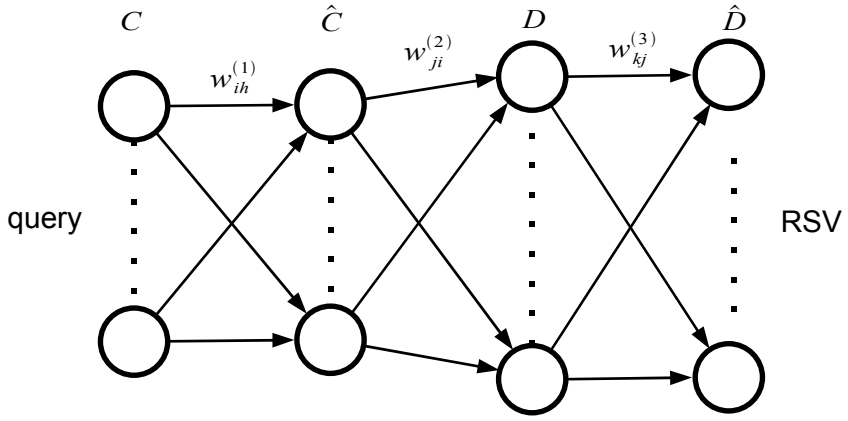

Figure 5.1: The associative network proposed by Scheir [2008]. The network implements a feed-forward topology comprising four node layers ($C, \hat{C}, D$ and $\hat{D}$) referring to the concept, expanded concept, resultset and expanded resultset layer, respectively. The first weight layer $\mathbf{w}^{(1)}$ is based on domain knowledge stemming from a domain ontology. The second weight layer $\mathbf{w}^{(2)}$ is based on manual indexing and an inverse document frequency weighting scheme. The third weight layer $\mathbf{w}^{(3)}$ is based on textual similarity of documents (VSM).

point of view, the model comprises two node layers (see Figure 1.1). The units in the input layer represent concepts (i.e. controlled index terms) and the units in the output layer represent documents. The directed inter-layer connections are defined by domain experts in a manual indexing process. The final activation level of the document units is interpreted as the retrieval status value. Subsequently, documents are ranked in decreasing order according to their activation level. Furthermore, Scheir [2008] introduces intra-layer connections between concepts and documents, respectively. However, the resulting network is not intended to be recurrent, rather it resembles the feed-forward architecture presented in Section 4.1.1. This design decision eases computation since spreading activation in a feed-forward network is rather trivial compared to a recurrent network such as the one illustrated in Figure 1.1. The actual topology of the associative network is depicted in Figure 5.1. The query processing is defined as follows:

1. A query is issued comprising a number of concept nodes along with their initial activation values.

2. The activation spreads from the input layer C to the next layer $\hat{C}$. The activation of the nodes in layer $\hat{C}$ is given by a weighted sum of the activation values in layer C. This corresponds to activating concept nodes that are associated with the initial query concepts. This phase can be thought of as (global) *query expansion* based on knowledge that is derived from the domain ontology.

3. The activation flow proceeds further from layer $\hat{C}$ to layer D, activating documents that are indexed with concepts from the expanded query.

4. Finally, the activation propagates from the activated documents in layer D to their associated documents in $\hat{D}$. Scheir [2008] refers to this phase as *resultset expansion*. The retrieval status value of a document is given by the activation level of the corresponding node in $\hat{D}$.

The inter-concept connection weights $\mathbf{w}^{(1)}$ are based on domain knowledge. Scheir [2008] experimented with various heuristics to extract this sub-symbolic knowledge from the domain ontology. Most of the heuristics utilize the structure of the ontology to derive a measure of the semantic similarity of two concepts. Among the various heuristics were semantic similarity measures based on the shortest path between two concepts and similarity measures based on vector space similarity (i.e. cosine similarity) where each relation in the ontology is represented as a dimension in the vector space. The applicability of certain heuristics depends on the nature of the ontology (e.g. whether taxonomic relations are prevalent).

The weights of the concept to document connections $\mathbf{w}^{(2)}$ are based on inverse document frequency by limiting the output of each unit to a constant (i.e. 1). The inter-document associations $\mathbf{w}^{(3)}$ are based on textual similarity. In particular, the distance (i.e. cosine similarity) of two documents in the vector space model (see Section 3.1.1) was used as a proxy for the similarity of the two documents.

In the following two specific heuristics are presented which were used in the system evaluation of the proposed system.

5.1.1 Inter-concept associations based on least common subsumer

For computing the semantic similarity of two ontological concepts in an ontology where the prominent features are taxonomic relations, [Scheir, 2008] uses a measure proposed by Wu and Palmer [1994]. The measure is based on the depth of a node in a taxonomic structure. That is, the shortest path between the root of the taxonomy and the given node. The measure is computed by taking the depth of the least common subsumer of the two concepts and scaling it by the sum of depths of the two concepts. The similarity is given by,

$$sim(c_1, c_2) = \frac{2 \cdot lcs(c_i, c_j)}{depth(c_i) + depth(c_j)} \tag{5.1}$$

where, c_i and c_j are the two concepts, $depth(c_i)$ is the shortest path between c_i and the root of the hierarchy and $lcs(c_i, c_j)$ is the depth of the concept with the longest path to the root that c_i and c_j share as an ancestor.

According to Scheir [2008], the major advantage of the above measure is that it addresses a typical problem of taxonomy-based similarity approaches, namely the more specific (i.e. the deeper in the hierarchy) a particular concept is, the more similar is the concept to its (immediate) predecessors and successors, respectively.

5.1.2 Inter-document association based on textual similarity

The similarity between two documents is computed using a modified cosine similarity measure based on a variant of the tf-idf weighting scheme. Scheir [2008] uses the scoring mechanism of the full-text retrieval engine Lucene[1] to compute the textual similarity between a document

[1] `http://lucene.apache.org` (last visited 29.11.2008)

d_i and a document d_j by extracting the 25 highest term weights from d_i and computing the similarity according to

$$sim(d_i, d_j) = score(\tilde{d_i}, d_j) \tag{5.2}$$

where $\tilde{d_i}$ is the vector representation of d_i with all but the 25 highest term weights zeroed out and $score$ is the default scoring function[2] of Lucene.

The rationale for "zeroing out" all but the 25 highest term weights is the fact that query time is proportional to the size of the query. The number of non-zero components of d_i is potentionally large hence the transformation of d_i to $\tilde{d_i}$ can be regarded as a performance/accuracy tradeoff.

5.2 A Connectionist Model for the APOSDLE Platform

The goal of this thesis is to complement the associative retrieval model presented above with a learning procedure that adapts the associative network based on explicit feedback provided by the user. As pointed out above, the topology of the associative network bears similarities with a feed-forward neural network. It follows that the error-backpropagation algorithm, presented in Section 4.1.2, is a natural choice for the learning procedure. Over more than two decades, the backpropagation algorithm has been subject to a vast amount of research and is thus very well understood - both, theoretically and empirically. Due to this fact and the efficiency of the algorithm that has already been pointed out in Section 4.1.2, the author favored the backpropagation algorithm over other connectionist learning procedures which have been successfully used in IR [Belew, 1989; Kwok, 1995].

In the following, the major contribution of this thesis, a connectionist retrieval model based on a backpropagation network is presented. The proposed connectionist model employs ideas from probabilistic indexing[3] by regarding the activation of a document node k in the output layer as an estimate of the *probability of relevance* of document d_k with respect to the query. To implement the ideas from probabilistic indexing presented above, the following modifications to the associative retrieval model are proposed:

- The activation level of a node in the layer $\hat{D}$ is regarded as an estimate of the probability of relevance of the corresponding document w.r.t. the query. To constrain the activation level of a document node to the interval $[0, 1]$ the logistic sigmoid function given by (4.5) is used.
- The relevance feedback data provided by the user is mapped onto an estimate of the probability of relevance of that document w.r.t. the query. In the course of this thesis only the case of binary relevance is regarded. In which the fact that a document is relevant is mapped onto a probability estimate of 1, and non-relevant is mapped onto an estimate of 0.

Based on these modifications each output unit (i.e. each document node in $\hat{D}$) can be regarded as a binary classification problem, where the k-th unit models the probability of relevance $P(R = 1|d_k, q)$ of document d_k. As promoted in Section 4.1.2, the cross-entropy error function given by (4.8) can be used to fit the model parameters to maximize the (log) likelihood

[2] `http://lucene.apache.org/java/docs/scoring.html` (last visited 29.11.2008)

[3] See Section 3.2 in general and Section 3.2.2 in particular.

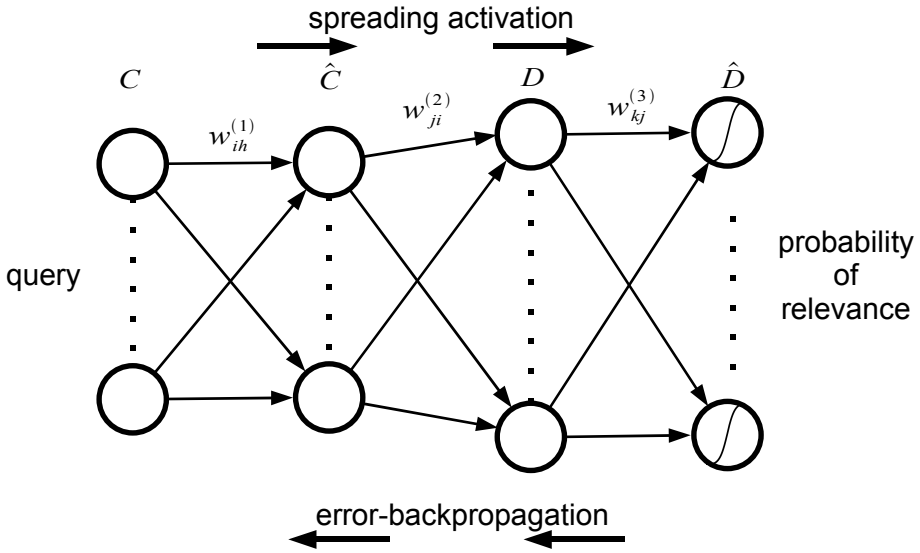

Figure 5.2: The connectionist retrieval model based on the associative retrieval model introduced by Scheir [2008]. The associative network is modified by introducing logistic sigmoidal activation functions in the output layer. Hence, the activation of the output units can be regarded as an estimate of the probability of relevance of a particular document given the query.

of the training data similar to the probabilistic indexing approach presented in Section 3.2.2. The resulting model is shown in Figure 5.2. The arrows indicate the direction of information processing during retrieval and network training, respectively.

As already pointed out, traditional probabilistic indexing approaches fail due to insufficient training data because too many model parameters need to be estimated (cf. [Fuhr, 1992]). However, because of the initialization of the associative network based on the heuristics outlined above, the ”untrained” model has already desirable retrieval capabilities. This initialization can be regarded as a reasonable initial estimate of the model parameters which we would like to further improve based on relevance feedback. The Probabilistic Indexing and Retrieval Component System (PIRCS) by Kwok [1995] described in Section 4.2.2 uses a similar strategy by utilizing (established) IR heuristics to ”bootstrap” the feed-forward network underlying PIRCS in order to overcome the problem of insufficient training data. Schütze et al. [1995] also point out that this approach has potential for enhancing the effectiveness of IR models based on neural networks. However, in contrast to our work, Kwok [1995] does not use the powerful backpropagation algorithm for network training.

Another difference between PIRCS and the proposed model is the fact that PIRCS only considers indexing term weights. It has already been pointed out that the first weight layer in the associative network implements a form of query expansion. We might regard it as a transformation from the query concept space to the document concept space. Thus, during network training, the first weight layer acts like a linear version of a transformation network trained by the total learning strategy presented in Section 4.2.3. Due to the fact that the transformation network presented by Crestani and van Rijsbergen [1997] is implemented using a two-layer neural network (i.e. one hidden layer), the network weights cannot be initialized using heuristics, such

as term cooccurrence or domain knowledge in form of a thesaurus or an ontology, to overcome the insufficient training data problem.

5.2.1 Network Training

The network is trained using stochastic gradient descent. This enables an on-line training of the network, a query at a time. As pointed out in Section 1.2, on-line training is a functional requirement of the model. It allows the instant adaption of the network with respect to the user feedback. However, the concrete gradient-based optimization algorithm is independent of the model proposed in this section. We might as well employ a batch learning method such as the conjugate gradients method mentioned in Section 4.1.2. Another advantage of stochastic gradient descent is that there is no need to store the training examples after they have been processed. The benefit of this is not only storage efficiency, due to anonymity policies retaining training data over a long period of time might even not be possible.

The gradient with respect to the network parameters is computed using the error-backpropagation algorithm. However, in contrast to the general case described in Section 4.1.2, in this setting we usually do not have a target value for each output unit. In general, the user does not provide the relevance assessments for the complete ranking but rather for a small subset thereof. Hence, the computation of the errors differs from Section 4.1.2 since we can only compute an error if we have the corresponding target value.

For a given input pattern q and a set of corresponding relevance judgements $\mathcal{J}$ network training proceeds as follows:

1. The error or the k-th output unit δ_k is computed according to,

$$\delta_k = \begin{cases} y_k - t_k, \text{if } k \in \mathcal{J} \\ 0, \text{otherwise} \end{cases} \tag{5.3}$$

   where y_k is the activation of document node k in layer $\hat{D}$ transformed by the logistic sigmoid function and $t_k \in \{0, 1\}$ is the binary relevance judgement of document k w.r.t. q.

2. The δ_k's are propagated back as described in Section 4.1.2 to give the values of δ_j of the preceding node layer. since the activation function $h(a)$ of the layers D and $\hat{C}$, however, is linear, Equation 4.17 reduces to,

$$\delta_j = \sum_k w_{kj}\delta_k \tag{5.4}$$

   where w_{kj} is the weight of the connection from node j to node k. The value δ_j of a unit j in the current layer is computed based on the (weighted) error of each unit k in the proceeding node layer (i.e. the δ_j values of layer D are the values of δ_k for computing the errors of layer $\hat{C}$).

3. The gradient w.r.t. each connection weight is evaluated. Again we make use of the simplifying fact that the output of a unit in layers $C, \hat{C}$ and D is the same as the input of the unit, $z_i = a_i$. Thus, the gradient is given by,

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j a_i \tag{5.5}$$

where again w_{ji} denotes the weight of the connection from node i to node j and E_n denotes the cross-entropy error function given by (4.8).

4. Finally, each connection weight w_{ji} where $\frac{\partial E_n}{\partial w_{ji}} \neq 0$ is updated according to,

$$w_{ji}^{(t+1)} = w_{ji}^{(t)} - \eta \cdot \frac{\partial E_n}{\partial w_{ji}} \tag{5.6}$$

where η is the learning rate that defines the step size in the parameter space. If η is too small, the algorithm progresses slowly and a large number of epochs is needed for convergence. On the other hand, if η is too large, the algorithm might "jump over" (local) minima in the error function or might even start oscillating around a minimum hence making no progress at all. Note that the superscripts $t+1$ and t refer to the time (new and old weight, respectively) and not to the weight layer.

The above procedure is applied to each query in the learning sample and the whole process is repeated until a predefined stopping criterion is met (e.g. convergence, a fixed number of iterations, error on held-out data increases). The learning procedure outlined above is generic, meaning that it is not limited to the topology depicted in Figure 5.2 but can be used for different feed-forward topologies with any number of weight layers.

In Section 4.2, it has been pointed out that - besides the presence of a learning procedure - the presence of any non-linearities (i.e. non-linear activation functions) is what distinguishes neural networks from spreading activation. However, our model does not employ non-linearities in the hidden units. As a consequence, the function[4] represented by the above network comprises a number of linear transformations. Bishop [2006] points out that,

> *If the activation functions of all the hidden units in a network are taken to be linear, then for any such network we can always find an equivalent network without hidden units. This follows from the fact that the composition of successive linear transformations is itself a linear transformation.*

It follows, that retrieval capabilities of the network might as well be implemented using a single layer network. Hence, from a learning perspective, the multi-layer representation of the associative retrieval model is inefficient because it more than doubles the amount of adjustable parameters in the model without increasing the capacity of the learning method. Limited capacity means that linear networks can only represent linear decision boundaries (as shown in Figure 3.5). In general, being limited to a linear decision boundary is not necessarily a disadvantage. Especially in the text domain, which is usually governed by large feature spaces and rather small training sets, compared to the dimensionality of the feature space, choosing a *biased* classifier (i.e. a linear one) might even result in increased effectiveness since linear classifiers are in general more robust with respect to noisy (high variance) data[5]. In fact, experiments conducted by Schütze et al. [1995] showed that, in the context of information filtering, non-linear neural networks do not provide any additional benefit over linear neural networks, which is contrasted by the fact that they are much harder to train.

[4] In Section 4.1.1 we noted that a neural network can be considered as a non-linear function mapping from some input space to an output space governed by a vector of adjustable parameters (cf. [Bishop, 2006]).

[5] For detailed information about this bias-variance tradeoff see [Manning et al., 2008].

5.2.2 Variants

Considering the associative network depicted in Figure 5.1, we observe that the different weight layers exhibit different levels of sparsity. In particular, the concept to document ($\mathbf{w}^{(2)}$) layer is relatively sparse compared to the document to document ($\mathbf{w}^{(3)}$) layer (i.e. most of the possible connection weights are zero). This sparsity is due to the fact that the human indexers are restricted to binary indexing decisions. This is certainly not desirable and limits the retrieval capabilities of the system. Hence, we would like to use as much probability mass (i.e. training data) as possible to estimate these parameters. Based on this observation, we might restrict the generic learning algorithm to certain layers. In the following two variants are proposed. Both use the associative network as outlined above for forward propagation (spreading activation) however they restrict learning (adaption of connection weights) to certain layers.

Backprop C-$\hat{C}$-D The use of cosine similarity in the vector-space model is an established means to estimate the similarity of documents (or any textual data) and its effectiveness has been proven in many experiments and real life applications. Therefore, we might conclude that the third weight layer $\mathbf{w}^{(3)}$ is properly initialized and we can use the evidence in our training data to focus learning on the other weight layers. Another fact why we would like to disregard the third weight layer during learning is that in general many more nodes in the node layer D are "active" compared to the preceding node layers C and $\hat{C}$. Thus, in general the gradients w.r.t. the weights in layer $\mathbf{w}^{(3)}$ are much more likely to be non zero than the gradients w.r.t. $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$, respectively. Again, because of the density of the third weight layer, this causes a fine grained distribution of the errors among the connections in $\mathbf{w}^{(3)}$. Because of the noisy training data - relevance feedback data is ambiguous, which is the consequence of the fact that RF is usually an expression of *subjective relevance*[6] - the learned connection weights are very unstable. For this reason, much more training examples are needed to maintain the generalization capabilities of the network. Concluding that a proper training of the third weight layer is hard and requires a lot of training data we might make the best out of the available data by restricting the learning process to the first two weight layers. For this purpose we compute the RSVs and δ_k's, as usual, by spreading activation through the network depicted in Figure 5.1 and computing the error w.r.t. the relevance judgements, respectively. However, we alter the backpropagation procedure to compute the value of δ_j in the node layer $\hat{C}$ not w.r.t. the values of δ_k in D but w.r.t. the value δ_k of the corresponding node in $\hat{D}$.

Backprop $\hat{C}$-D By restricting the adaption of the network solely to the second weight layer we employ a notion of automated indexing. The δ_k's are computed as usual (using the activation level in $\hat{D}$ as the network output) but the gradients w.r.t. the weights in the second layer are simply the activation level of the source node in $\hat{C}$ multiplied by the error of the corresponding target node in $\hat{D}$. Note that since we only regard a single weight layer this strategy is essentially the application of the delta-rule.

$$w_{ji}^{(t+1)} = w_{ji}^{(t)} - \eta(y_j - t_j)a_i \tag{5.7}$$

[6] See Section 2.1.3

5.3 Implementation

The connectionist model presented above has been implemented as a component of the APOSDLE platform[7] [Lindstaedt and Mayer, 2006]. The APOSDLE platform features a service-oriented architecture (SOA). Services, such as the retrieval service, are exposed as web-services using the Simple Object Access Protocol (SOAP). The platform is written in the Java programming language, it is highly modular and promotes extensibility. The configuration and deployment of the platform is realized by using a dependency injection container provided by Spring[8]. This enables the "wiring" of software components without the need to write Java code. Dependency injection (as a form of *inversion of control*) encourages loose coupling of software components and facilitates component reuse [Fowler, 2004]. The reference implementation of the model builds upon the associative network component created by Scheir [2008]. However, due to slightly different requirements a number of modifications have been made. The following section is structured into two parts: First, the requirements of the data structure representing the associative network are outlined and the realization of the data structure is described. The second part is dedicated to the software architecture of the reference implementing the connectionist model presented above.

5.3.1 Graph data structure

In the associative retrieval model proposed by Scheir [2008] the graph representing the associative network is sparse and access to individual edges is not necessary since all adjacent nodes to an active node have to be processed. It follows that an adjacency list representation is favored to an adjacency matrix representation (cf. [Kleinberg and Tardos, 2005]).

In the reference implementation in the APOSDLE platform the adjacency list representation of the associative network is stored in a relational database. For each of the three edge types (concept to concept, concept to document and document to document) there is a separate table which contains a tuple for each edge. Each table has three attributes: the Uniform Resource Identifier[9] (URI) of the tail, the URI of the head and the weight. The data types for the first two attributes is a variable-length string of up to 500 characters. The character encoding is latin-1, a single-byte character set. The type of the weight attribute is a single precision floating point number. The total amount of bytes required to represent a single edge is $2(1+l_a)+4 = 2l_a+6$, where an additional byte is used to represent the length of each string and l_a denotes the average length of the URI strings of the nodes.

The first attribute of each table (i.e. the tail of an edge) is indexed using a B-Tree data structure [Garcia-Molina et al., 2008, p.633ff]. According to [Sedgewick, 2003] this database representation is equivalent to an adjacency list indexed with a B-Tree (cf. [Scheir, 2008]). The primary index determines the physical structure of the table. It follows, that tuples with the same tail node are sequentially placed in the file system (i.e. in the same or adjacent blocks of the file system), enabling sequential access to the adjacency lists. Ensuring sequential access is crucial for efficient serving of retrieval requests since response time is governed by access to disk.

In contrast to the associative retrieval model, the backpropagation procedure needs to access

[7] See Section 1.2.

[8] `http://www.springframework.org/` (last visited 26.11.2008)

[9] For detailed information see [Berners-Lee et al., 2005]

individual edges. Accessing an edge in the graph requires a scan of the adjacency list of the tail of the edge. In the worst case this takes time proportional to the degree of the node $0(d_n)$. However, due to the fact that the major bottleneck in accessing the data structure is the access to disk, the time it takes to traverse the list (which takes place in memory) can be neglected.

One requirement of the implementation of the network training module was that the adjacency list representation of the original associative network should be immutable. Rather, the modifications during network training should be represented in an overlay network which defines the bias of each edge in the original associative network. The overlay network is stored analogous to the immutable network in three separate relations. This design decision enables the switching between the original (untrained) and the trained model during operational mode. Disk seeks and space requirements, however, are potentially doubled. "Potentially" refers to the facts that the overlay network is sparse and that the relations in the bias tables are indexed using both, the head and tail attribute. In general, the B-Tree which is used as the index data structure is compact enough to fit into main memory, allowing to check whether or not an edge bias exists without the need to actually access the disk.

5.3.2 Software architecture

In the following the software architecture of the associative network module of the APOSDLE platform is described. The reference implementation of the connectionist model presented in Section 5.2 is part of this module. The important aspects[10] of the architecture are communicated using the Unified Modeling Language (UML) [Fowler, 2003].

The package diagram shown in Figure 5.3 depicts the logical structure of the module. The source code is organized into four sub-packages: `services`, `database`, `topology` and `learning`.

The `database` package contains the low-level view on the graph data structure. Its classes simply wrap the relational database tables to allow coherent and convenient access to the graph. The `topology` package presents a high-level abstraction of the graph data structure by providing classes which enable the spreading of activation between adjacent node layers. The `learning` package contains classes that implement different strategies to update the connectionist representation (i.e. the associative network) based on relevance judgements. Finally, the `services` package defines the services provided by the module.

The core of the module is a software component which exposes two services:

`getKnowledgeArtefacts` This is the basic retrieval service provided by the APOSDLE platform. Given a query containing a number of domain model elements (concepts) the service returns a list of knowledge artefacts along with their RSVs. An invocation of this service causes a forward propagation through the associative network.

`submitFeedback` This service triggers the network training for a single query. The service requires the initial query and a list of relevance judgements and modifies the structure of the network based on these inputs.

The implementation of the two services is encapsulated in the `topology` and `learning` packages. Both packages are described in-depth in the reminder of this section. Sequence

[10] The reader should bear in mind that in order to maintain readability, all but the most important information is suppressed. Therefore, the author urges the reader not to infer anything by its absence.

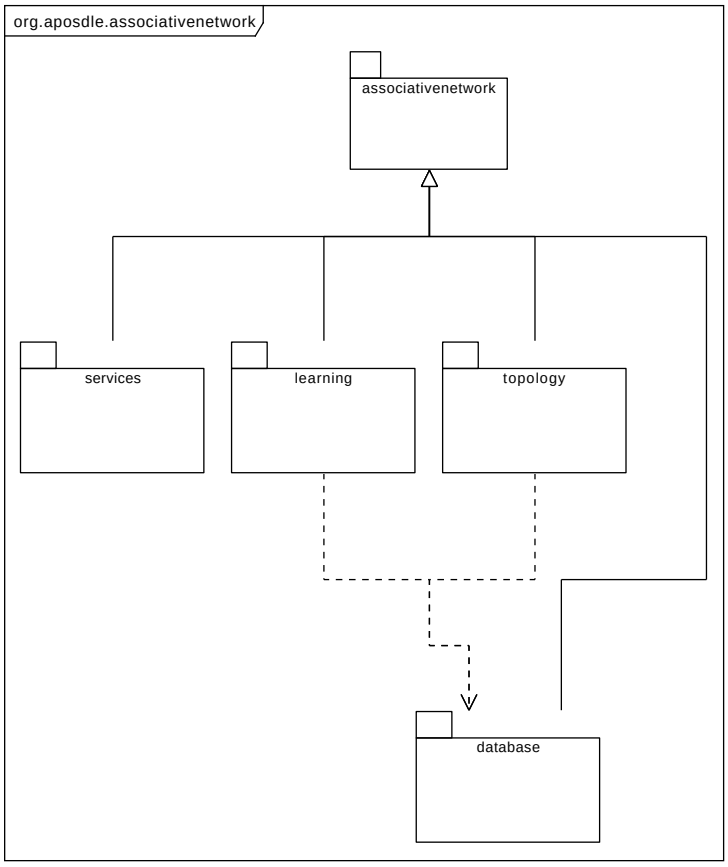

Figure 5.3: An UML package diagram showing the organization of the associative network module

diagrams showing the collaborations among objects in order to serve the above services are deferred to the in-depth discussion of the corresponding package. The software component `ConnectionistLearningANCompImpl` and its class dependencies are depicted in the class diagram shown in Figure 5.4. The component is configured using the setter injection functionality provided by the dependency injection container. An example configuration file is shown in Listing 7.1 in Appendix A. The container automatically resolves all object dependencies by injecting the instantiated objects using the setter methods provided by the component.

Services The `services` package contains interfaces that define services provided by the module. The organization of service interfaces into a separate package is a coding convention of the APOSDLE platform. The package comprises two service interfaces:

`AssociativeNetworkService` which exposes associative retrieval services.

`ConnectionistLearningANService` which extends the former service with the ability to submit feedback to the ranking of an initial query.

Both services are depicted in the class diagram in Figure 5.4.

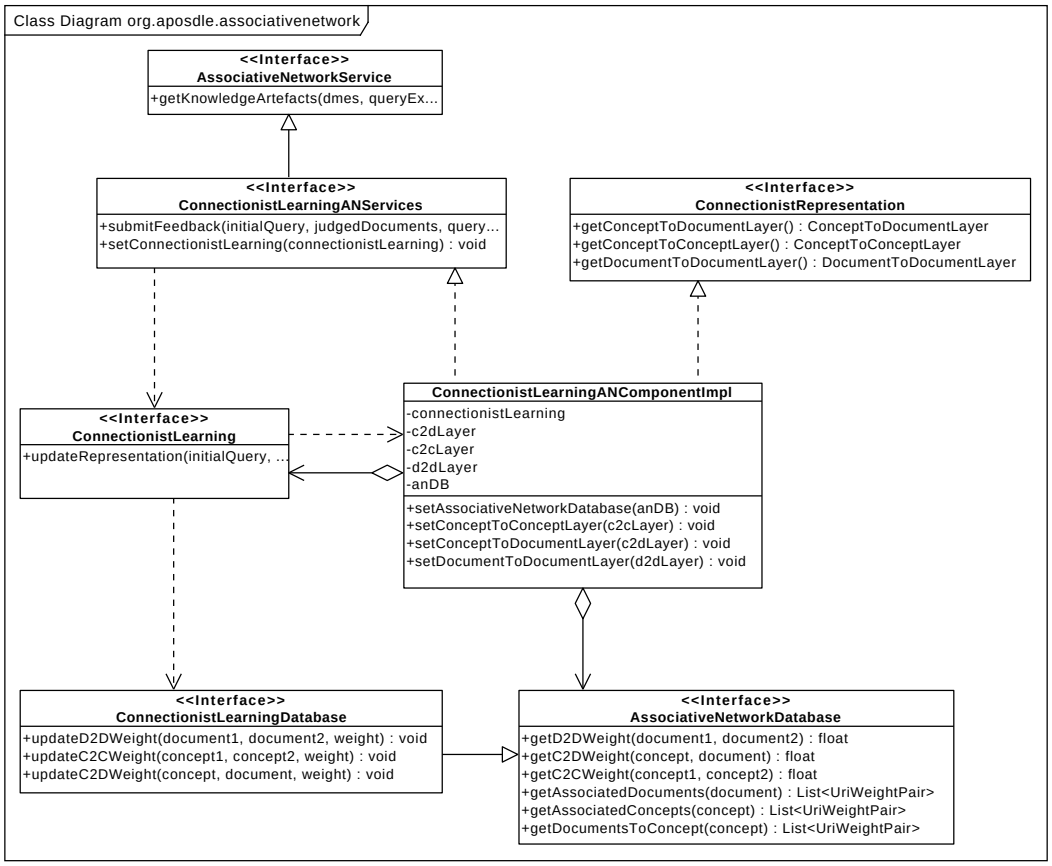

Figure 5.4: An UML class digram showing the `ConnectionistLearningANCompImpl` class and its dependencies.

Database The `database` package contains classes that wrap the database representation of the associative network. There are two basic interfaces: `AssociativeNetworkDatabase` and `ConnectionistLearningDatabase`. The first interface specifies methods for finding associated concepts and documents given a concept or document as well as methods to access individual edges. The second interface extends the first by specifying methods to update edges. For each of the above interfaces there is a concrete implementation wrapping the corresponding database tables. Both, `AssociativeNetworkDatabaseImpl` and `ConnectionistLearningDatabaseImpl` use the Java Database Connectivity (JDBC) API abstraction provided by Spring. `ConnectionistLearningDatabaseImpl` implements only the update functionality and delegates to a concrete `AssociativeNetworkDatabase` class for reading (retrieving individual edge weights).

Network training is a computational intensive process, each input pattern requires a forward propagation and afterwards individual edge weights have to be accessed and possibly updated. Since access to the graph requires access to disk, each input pattern requires multiple disk seeks. In order to speed up the process, simple caching strategies have been implemented which are based on the fact that the original associative network is immutable. The class `AssociativeNetworkDatabaseCache` wraps a concrete `AssociativeNetworkDatabase` class and caches adjacency lists in a `WeakHashMap` data structure. The use of a `WeakHashMap` enables the Java Virtual Machine (JVM) to garbage

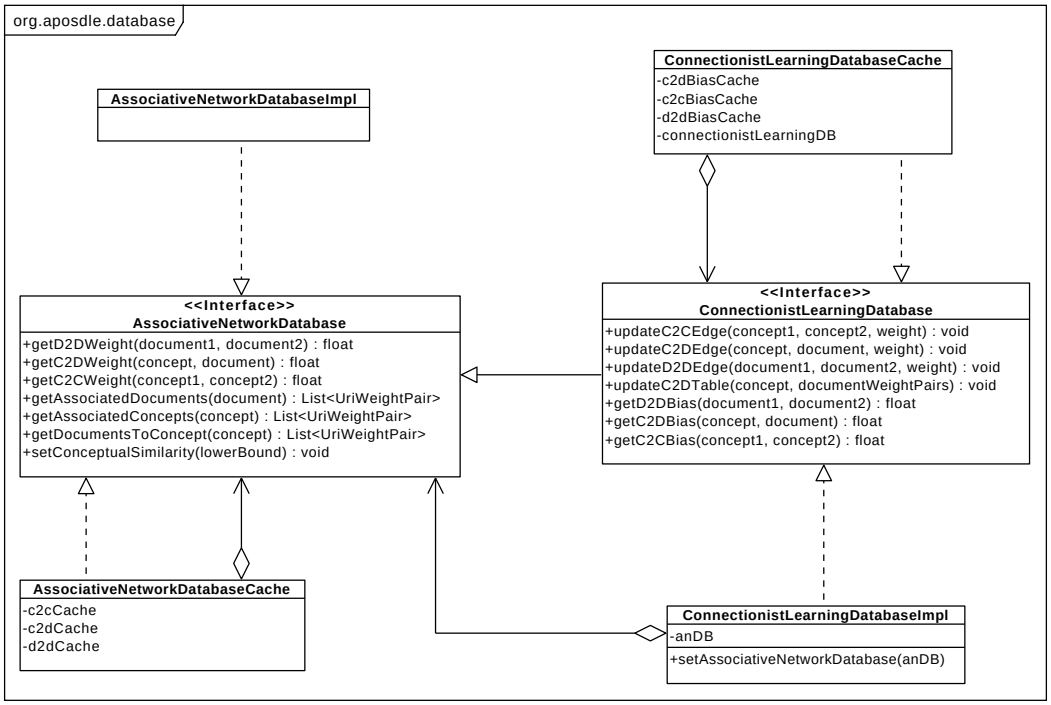

Figure 5.5: An UML class digram showing the `database` package and the dependencies among its classes.

collect cached adjacency lists if it runs out of heap space.

The `ConnectionistLearningDatabaseCache` is implemented analogously but due to the fact that the `ConnectionistLearningDatabase` is not immutable, it has to implement a write policy to ensure coherency. Currently, the class implements a *no-write allocation* cache policy, which means that writes are not written back to the database. This is certainly not adequate for productive code but is sufficient for evaluation purpose. The classes in the `database` package and their dependencies are shown in Figure 5.5.

Topology The `topology` package represents an abstraction of the low-level graph representation in the `database` package. It comprises abstractions for the weight layers of the associative network. The package contains three interfaces representing the three different weight layers:

- `ConceptToConceptLayer`
- `ConceptToDocumentLayer`
- `DocumentToDocumentLayer`

Concrete classes implementing these interfaces implement the actual spreading activation functionality of the associative retrieval model. For detailed information about concrete implementations see Scheir [2008]. The interfaces provide methods with common signatures which activate nodes in the target layer given a node in the source layer. In the original associative retrieval model the nodes in the associative network implement a linear activation function. The

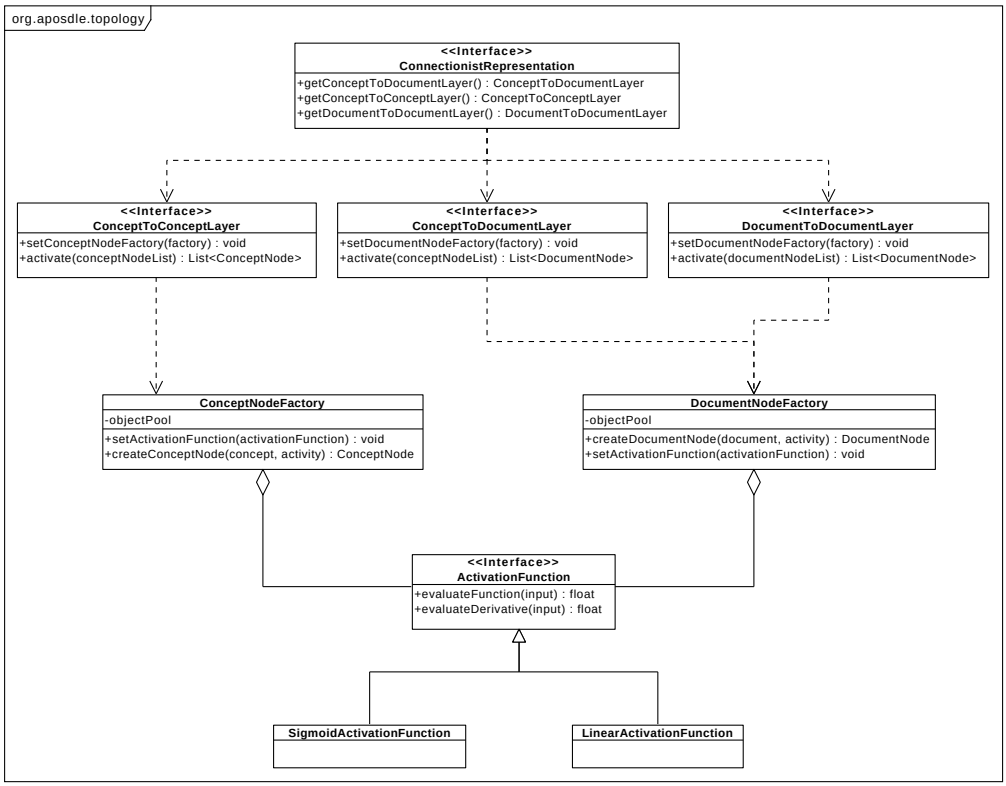

Figure 5.6: An UML class diagram of the topology package.

major modifications to this package comprise the introduction of arbitrary activation functions and factories managing the creation of document and concept nodes, respectively. The latter enables object reuse using the *flyweight* pattern [Gamma et al., 1995]. Object reuse is performance critical since object creation is costly (cf. [Bloch, 2008]). The major interfaces and classes in the package are depicted in Figure 5.6.

Concrete classes implementing the above interface are responsible for serving retrieval requests. The sequence diagram in Figure 5.7 shows the collaboration of objects implementing the above interfaces when serving a `getKnowledgeArtefacts` request.

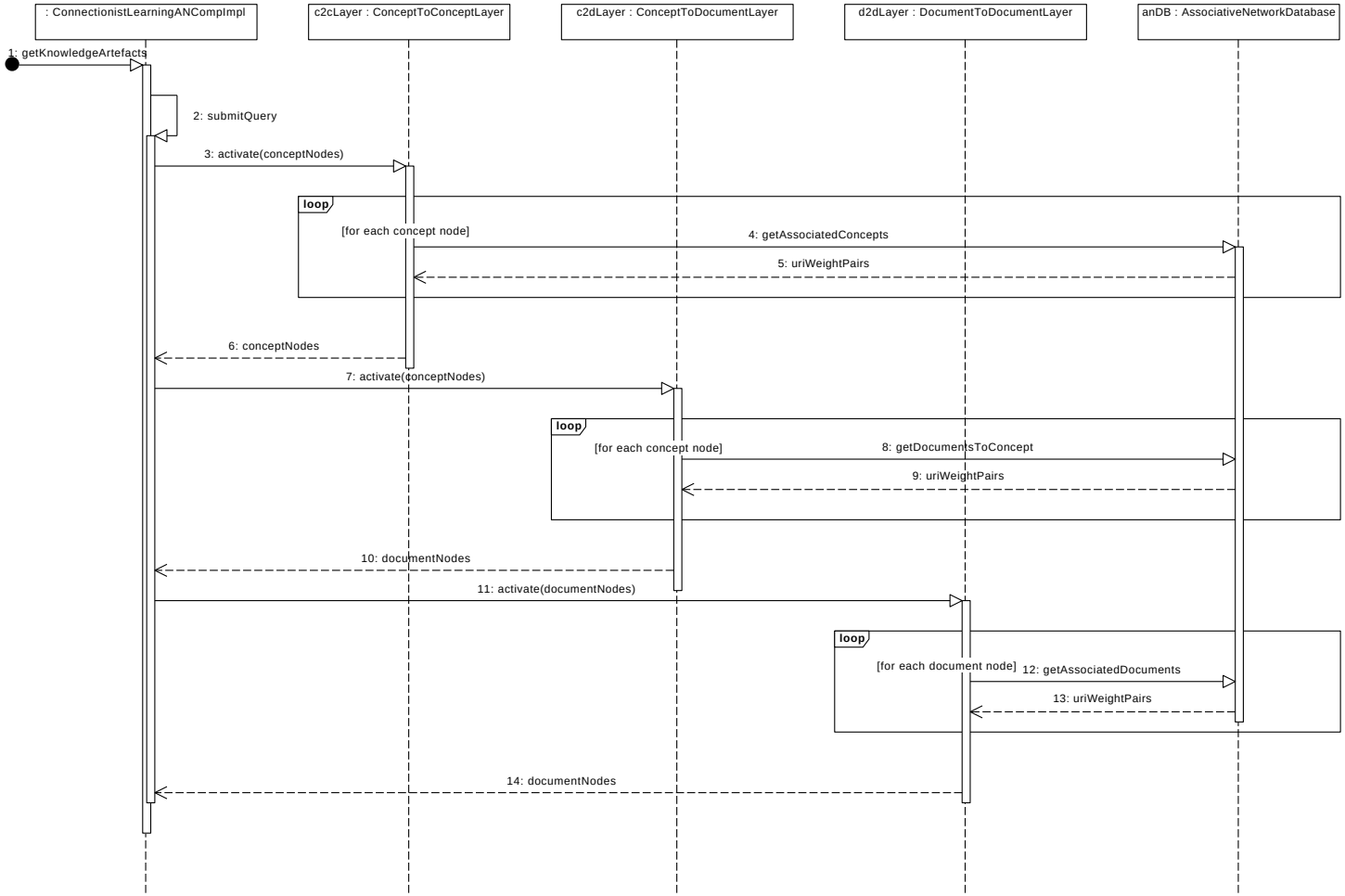

Figure 5.7: An UML sequence diagram showing the collaborations among objects while serving a `getKnowledgeArtefacts` request.

Learning The learning package contains the implementations of the learning algorithm presented in Section 5.2. The implementation follows the *strategy* design pattern [Gamma et al., 1995]. Each concrete learning strategy has to implement the `ConnectionistLearning` interface which provides a method to update a `ConnectionistRepresentation` given an initial query and a list of relevance judgements. Currently, three concrete implementations of the interface are available referring to the generic learning algorithm and the two variants. Figure 5.8 shows the class diagram of the `learning` package.

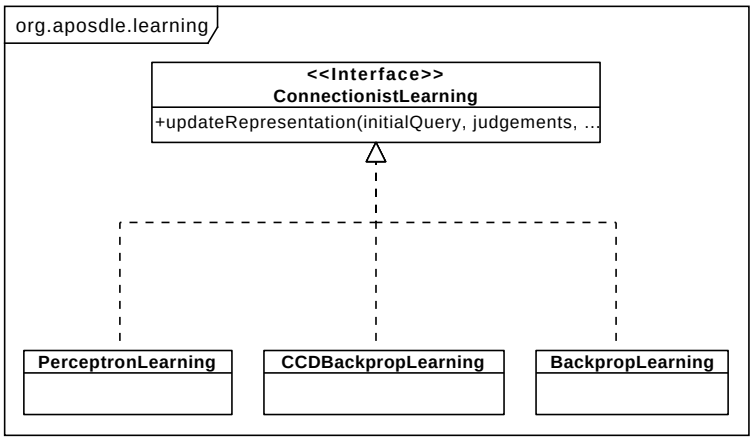

Figure 5.8: An UML class diagram of the `learning` package.

BackpropLearning refers to a concrete class implementing the generic learning algorithm as described in Section 5.2. The interaction of this class with other classes in the module is shown in the sequence diagram in Figure 7.1 in Appendix A.

CCDBackpropLearning implements the first variant introduced in Section 5.2.2. The learning procedure applies the backpropagation learning procedure to the first and second weight layer. The collaboration of the classes involved in the `updateRepresentation` procedure is similar to the sequence diagram in Figure 7.1 in Appendix A. However, the first and third loops are omitted.

PerceptronLearning is an implementation of the second variant of the generic learning algorithm. The name perceptron is a misnomer, rather the class implements a logistic regression model trained via stochastic gradient descent. Weight adaption is implemented by an application of the delta-rule. Figure 7.2 in Appendix A shows the collaborations among the classes involved in the `updateRepresentation` procedure.

5.4 System Evaluation

To evaluate the effectiveness of the connectionist model, a traditional IR experiment based on a test collection was conducted. The goal of the system evaluation is to provide preliminary evidence, whether or not the application of the learning procedure described above is practicable given the associative network proposed by Scheir [2008]. For this purpose, two approaches to system evaluation of interactive retrieval systems have been considered:

Residual Ranking , a method which evaluates the effectiveness of the system based on all documents which have not been used for feedback (the residual collection). The method is described in more detail in Section 2.3.4. The method is intended to be used to compare the effectiveness of different feedback techniques. Therefore, it can only be used to evaluate the effectiveness of the different feedback variants presented in Section 5.2.2 against the generic approach but not to compare the feedback approach with the baseline.

Train-Test Set , this is the evaluation method which is usually applied in the area of pattern recognition. The ground truth is split into a training and a test set. Subsequently, a model is learned based on the training data which is then tested on the test set. This method evaluates the generalization capabilities of the model because it evaluates the effectiveness of the model on queries which have not been seen during training.

The above approaches evaluate quite different things. The residual ranking method evaluates the effectiveness of the system on a per query basis. This means, that for each query some evidence is given and the method evaluates how the system uses this evidence to increase retrieval effectiveness on the residual collection of this query. Whereas the train-test set method evaluates how well the system uses the evidence to increase the effectiveness of other queries.

We chose the second method for the experiment due to the fact that our primary goal is to evaluate the connectionist model relative to the baseline. The train-test set method is usually performed using the cross-validation technique. For this purpose, the whole data set is partitioned into k partitions and each partition i is used once as the test set while all other partitions but partition i comprise the training set. The size of k is a tradeoff between computational costs and the amount of data that is used for training. In general, one wants to use as much data as possible for training since the process of labelling the data is costly. The largest amount of training data is achieved when choosing k to be the number of data points in the data set. This special case of cross-validation is referred to as *leave-one-out cross-validation* and was used in the experiment described in this section.

5.4.1 Test Collection

The effectiveness of the system was evaluated using a subset of the test collection created by Scheir [2008]. The original test collection was exclusively created by Scheir [2008] to evaluate his associative network model and has been used in subsequent experiments in the context of the APOSDLE platform [Scheir et al., 2007a,b]. It is based on the domain model and document corpus of the first release of the APOSDLE system [Lindstaedt and Mayer, 2006]. The corpus contains 1016 documents related to the Requirements Engineering domain. The domain ontology comprises 70 concepts. 496 documents in the corpus have been annotated by domain experts using (only) 21 of the available concepts provided by the controlled vocabulary of the ontology.

In his experiments, Scheir [2008] used a set of 79 queries which were formed by sets of concepts stemming from the ontology. Scheir created a depth-30 pool based on 12 different configurations of the associative network model outlined in Section 5.1, exhibiting different measures of semantic similarity, different association thresholds and different operational modes (whether or not query expansion and/or resultset expansion is performed). The resulting pool contained 1932 query-document pairs that have been judged by a single person (i.e. the author of the test collection himself). Additional 855 relevance judgments were included in the collection stemming from runs that did not take part in the experiment reported by Scheir [2008]. According

to Scheir [2008], the size of the depth-100 pool for the same runs amounts in a total of 5393 query-document pairs, thus effectively judging 51.68% of a potential depth-100 pool.

The test collection that was used in the course of this thesis uses a subset of the 79 queries. The set contains 26 queries which are shown in Table 7.1 in Appendix A. In particular, only those single concept queries were considered for which Configuration-10 returned documents. Configuration-10 refers to the associative network as presented in Section 5.1 with semantic similarity based on the least common subsumer heuristic and a (semantic) similarity threshold of 0.7. The semantic similarity threshold is used to control the sparsity of the first weight layer. Depending on the employed measure of semantic similarity, the first weight layer is very dense (mostly non-zero), with the result that most of nodes in the second node layer are activated. This further leads to an overspreading of the network (almost all nodes in the associative network are activated) which harms retrieval performance.

In the following, the retrieval effectiveness of Configuration-10 will be referred to as the baseline. The retrieval effectiveness of the proposed extensions will be measured relative to this system configuration.

5.4.2 Evaluation Measures

In line with other experiments based on the same test collection [Scheir et al., 2007a; Scheir, 2008], inferred average precision (infAP)[11] was used as the target measure for determining retrieval effectiveness. Furthermore, the values of precision at various cuf-off levels (5,10,20,30) are also reported, although it has to be noted that there are too few queries in the test collection to draw reliable conclusions based on these scores.

5.4.3 Evaluation Setup

The associative network was initialized according to the baseline configuration. One run was performed using solely the initialized system without any learning at all. The results for this run were used as the baseline. As stated above, the system effectiveness of adaptive runs were measured based on leave-one-out cross-validation. For each of the 26 queries the network was trained on all but the query in question. Then the effectiveness of the system was measured on the single query that was not considered during training. After each fold, the network parameters were reset.

Early stopping based on a validation set should have been used to determine the optimal number of training epochs for each query and to avoid overfitting. But it was abandoned due to performance reasons. Rather, early stopping was performed on a subsample of (four) queries providing evidence that 30 epochs is a reasonable number to balance runtime and under-/overfitting. The learning rate parameter η was determined using the same subsample of queries. The initial value of $\eta = 0.01$ turned out to work best - lower values caused only minor changes in the ranking, whereas a larger value caused very unstable results and in general resulted in a weaker effectiveness on the subsample.

[11] See Section 2.2.2.

5.4.4 Evaluation Results

This section reports the results of the system evaluation that was conducted in the course of this thesis. The experimental setup is described in the preceding section. Three different runs of the extended system were submitted. The first, Backprop, refers to the generic learning algorithm described in Section 5.2. The second, Backprop C-$\hat{C}$-D, refers to the variant that restricts parameter adaption to the first and second weight layers. The third run, Backprop $\hat{C}$-D, refers to the second variant that restricts learning solely to the second weight layer. Table 5.1 shows

Measure	Baseline	Backprop $\hat{C}$-D	Backprop	Backprop C-$\hat{C}$-D
infAP	0.7609	0.7810 (+2.64%) ○	0.6933 (-8.88%)	**0.7974** (+4.80%) ○
Prec@5	0.8769	0.8923 (+1.76%)	0.6615 (-24.56%) ●	**0.9000** (+2.63%)
Prec@10	0.7885	0.8308 (+5.36%) ○	0.6154 (-21.95%) ●	**0.8385** (+6.34%) ○
Prec@20	0.6846	**0.7058** (+3.10%)	0.5308 (-22.46%) ●	0.7038 (+2.80%)
Prec@30	0.5808	**0.5846** (+0.65%)	0.4679 (-19.43%) ●	0.5782 (-0.45%)

○, ● statistically significant improvement or degradation

Table 5.1: Evaluation scores of the three runs and the baseline system. Statistically significant improvement or degradation over the baseline is based on a 95% confidence interval.

the evaluation results of the three runs compared to the baseline run. The empty and filled points refer to statistically significant improvement or degradation, respectively. For testing statistical significance, the author followed Sanderson and Zobel [2005] by choosing the dependent t-test together with a confidence interval of 95%. The dependent t-test is used when only one sample is available (the 26 queries) but we have repeated measurements (effectiveness of the baseline and the run, respectively). It assigns a confidence score to the null hypothesis, that is that both values are drawn from the same population. If the confidence in the null hypothesis (the p-value) is below a certain threshold (in our case ≤ 0.05) the null hypothesis is rejected. If the null hypothesis is rejected, we are able to report a statistically significant difference of the two runs [Sanderson and Zobel, 2005]. The implementation of the dependent t-test provided by [Jones et al., 2001] was used to determine the p-value given the per query effectiveness scores of the baseline and the extended model.

The results show that the (naive) generic approach performs worse than the baseline and the two variants. It is worse than the baseline on all effectiveness measures. However, the difference to the baseline in terms of infAP is not statistically significant. Table 7.1 in Appendix A shows detailed evaluation results for each query. When looking at the table one can see that the per query infAP scores of the generic approach (Backprop) are very instable. In particular, the effectiveness of the generic approach on topics 1,2,10,11,35,36 is superior to the other approaches. However, the effectiveness deteriorates on other topics. The author assumes that this is due to the facts that have already been pointed out in Section 5.2.2, namely that there is not sufficient training data available to train the document to document connections properly, following that the network completely overfits the training data and gives poor performance on the test data. Figure 5.9 shows a histogram which measures the infAP score of the Backprop run on each query against the infAP score of the baseline run on that query.

On the other hand, the two variants significantly outperform the baseline on the target measure and in terms of Prec@10. The relative performance increase in terms of infAP of Backprop

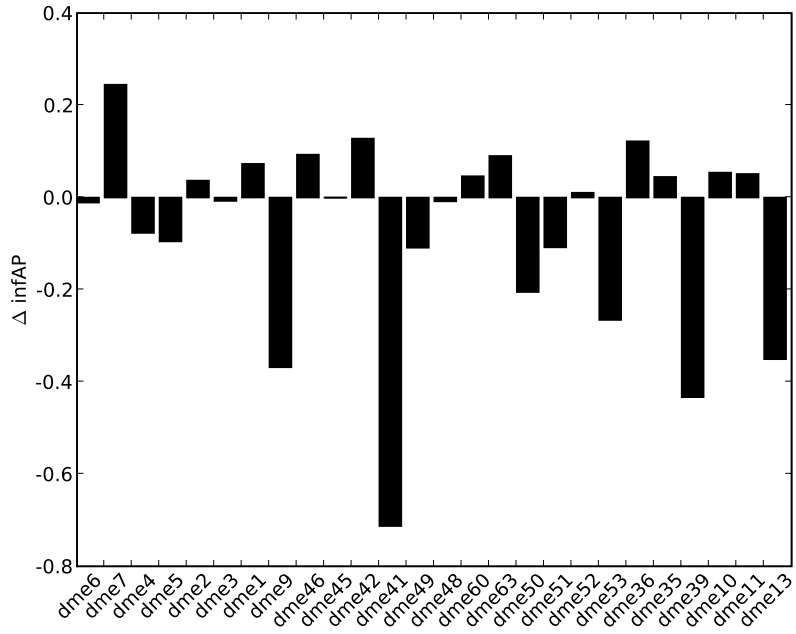

Figure 5.9: The relative difference in terms of infAP of the Backprop and the baseline runs. A positive Δ infAP indicates superior performance of Backprop.

C-$\hat{C}$-D over the baseline is 4.8%. A statistically significant relative increase in effectiveness of about 5% is very promising and can be considered as a positive, yet surprising, result. Figure 5.10 shows the difference histogram in terms of infAP between Backprop C-$\hat{C}$-D and the baseline on a per query basis. One can see that the extended model consistently outperforms the baseline on all but one topic. A reason for the poor performance of query 4 seems to be the already poor effectiveness of the initial query. Considering the baseline system, query 4 is by far the worst performing topic with an infAP score of 0.1813 (the average infAP score of the baseline is 0.7609).

The effectiveness of Backprop C-D is slightly worse than Backprop C-$\hat{C}$-D, the relative increase in effectiveness with respect to the baseline is 2.64% compared to 4.8% for Backprop C-$\hat{C}$-D. However, Backprop C-D turns out to be more robust than the latter. As can be seen in the infAP histogram in Figure 5.11, Backprop C-D does not fail miserably on any topic - the inferior effectiveness on topic 51 is hardly recognizable (below 0.03).

Summing up, the results of the system evaluation are very promising. Two of the three evaluated runs reported a significant increase in effectiveness on the target measure. Based on the outcome of the experiment we might conclude that the backpropagation algorithm is indeed an effective learning procedure to incorporate relevance feedback into the associative retrieval model proposed by [Scheir, 2008].

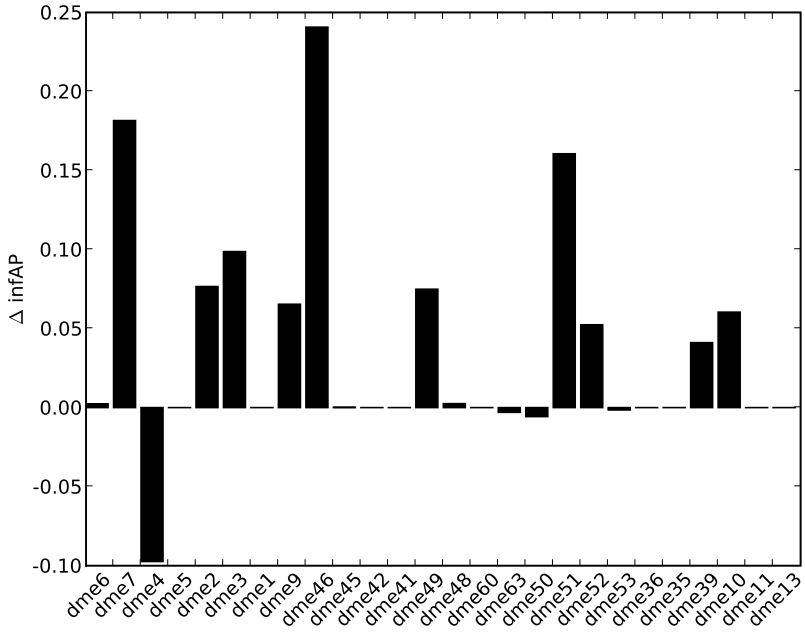

Figure 5.10: The relative difference in terms of infAP of the Backprop C-$\hat{C}$-D and the baseline runs. A positive Δ infAP indicates superior performance of Backprop C-$\hat{C}$-D.

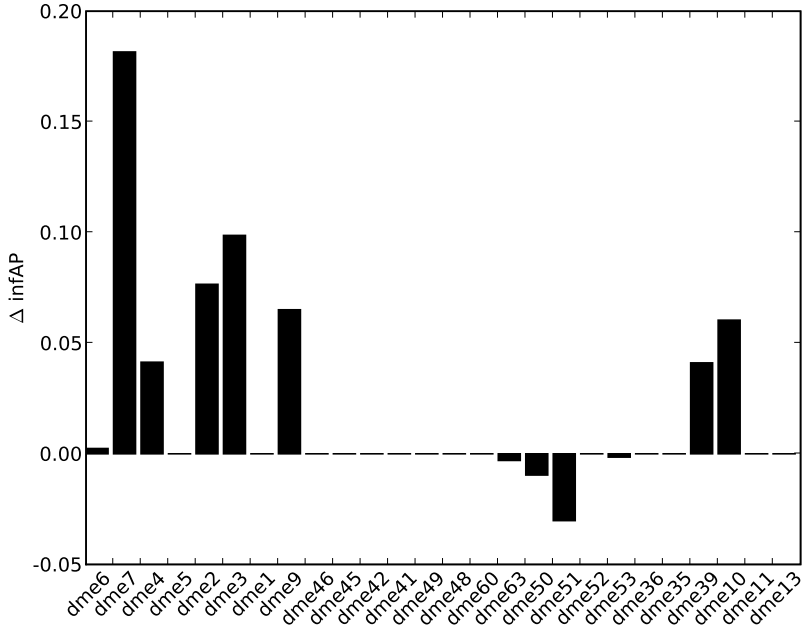

Figure 5.11: The relative difference in terms of infAP of the Backprop C-D and the baseline runs. A positive Δ infAP indicates superior performance of Backprop C-D.

Chapter 6

Conclusions and Outlook

6.1 Conclusions

6.1.1 Summary of Results

In the course of this thesis, a connectionist model for adaptive information retrieval has been developed. The proposed model is an extension of the associative retrieval model proposed by Scheir [2008] which is based on the spreading activation paradigm. It converges ideas from probabilistic indexing and backpropagation networks to improve the retrieval effectiveness based on feedback, received on prior performance. In particular, the model implements a document-focused learning paradigm which estimates the model parameters by regarding a single document to a number of queries. The model parameters are estimated by maximizing the (log) likelihood of the training data. To maximize the likelihood function, a simple gradient-based algorithm, known as stochastic gradient descent, is used. This optimization method enables an on-line adaption of the model - a query at a time - which was a functional requirement for the model. The gradient is evaluated using the efficient backpropagation algorithm, the classical learning procedure for feed-forward neural networks.

A reference implementation of the connectionist model has been created as a software component of the APOSDLE platform [Lindstaedt and Mayer, 2006]. Subsequently, the reference implementation has been evaluated using a traditional information retrieval experiment based on a small test collection of queries, documents and corresponding relevance judgements. In particular, the system evaluation focused on the ability of the model to generalize from feedback on different queries. That is, for each query in the collection, the network was trained on all but that query and the effectiveness of that query was reported. The results of the experiment show a significant increase in retrieval effectiveness of two variants of the proposed model compared to the baseline (i.e. the untrained model), providing preliminary evidence of the benefit of incorporating relevance feedback in the retrieval model proposed by Scheir [2008].

6.1.2 Critical Self-Assessment

In this section, a critical analysis of the reported results is given. The main point of critique of this thesis is the system evaluation in general and the test collection that was used in particular.

As stated in Section 2.2, the recommended number of topics to ensure reliability of results

is 50 - two times the size of the test collection that was used in the experiment conducted in the course of this thesis. Based on this fact, the reported results should be taken with a grain of salt. It has to be noted, however, that according to Sanderson and Zobel [2005] the acceptable minimum number of topics is 25, given that differences are statistically significant and a stable evaluation measure is used - both applies to the experiment described in Section 5.4. It follows, that the reported results capture at least to some extend the true ranking of the submitted runs. Nevertheless, to promote reliability, the topic set should have comprised at least 50 topics.

Another point of critique is the fact that the focus of the system evaluation is exclusively on generalization cabability of the model - that is, the improved performance on new queries. The goal of an adaptive retrieval model might as well be improved effectiveness on known queries. Again, it has to be noted that the evaluation of adaptive IR systems is an open research problem. To date, neither large-scale system evaluations of adaptive IR systems have been conducted nor is there any theoretical work on which kind of aspects the focus of such evaluations should be (cf. [Voorhees, 2008]).

6.2 Outlook

6.2.1 General Trends

This section points out two emergent trends in the context of (adaptive) information retrieval. The first trend is related to the current interest of the machine learning community in learning structured outputs in particular rankings. Since most problems in IR are ranking problems, this is of immediate importance to the IR community. In the past two years the ACM special interest group on information retrieval (SIGIR) dedicated two workshops at their annual conference to the topic of *learning to rank*. The second trend tackles the inherent problem of gathering sufficient training data. Currently, a lot of work in the context of relevance feedback has been dedicated to examine the use of implicit indicators (e.g. click-trough data) as a proxy for explicit relevance assessments [Joachims et al., 2005; Radlinski and Joachims, 2005; White et al., 2005, 2006].

Learning to Rank In contrast to the common approach in machine learning (and pattern recognition) in IR, the effectiveness measure that is used for parameter optimization during the training phase differs from the one that is used to test the final effectiveness of the system. Most parameter learning approaches in IR optimize the parameters w.r.t. the probability of relevance of a document given a query (cf. [Iyer et al., 2000]). Thus, those approaches rely on the Probabilistic Ranking Principle to ensure that those estimates indeed optimize the effectiveness of the system, measured in recall and precision. Early work on parameter optimization in IR such as the approaches presented in Section 3.2 viewed retrieval as either classification (binary relevance) or regression (graded relevance) [Wong and Yao, 1988; Fuhr and Buckley, 1991; Fuhr, 1992; Cooper et al., 1992; Nallapati, 2004]. The description-oriented approach [Fuhr, 1992] is the primary example for this *pointwise approach* where a model $\phi(q, d) \mapsto R$ is learned that maps from a joint feature mapping of a query and a document $\phi(q, d)$ to the class of relevant ($R = 1$) or non-relevant ($R = 0$) documents, resp., based on a learning sample of $\langle \phi(q_i, d_j), R \rangle$ tuples. In [Fuhr, 1992] the model is trained using standard loss functions for classification and regression such as cross-entropy and squared-error, respectively. However, the decrease in a pointwise error function (e.g. squared-error) does not necessary result in a better ranking (e.g.

measured in average precision). Why not directly optimize the parameters for the given ranking error function (e.g. average precision) instead of relying on the Probabilistic Ranking Principle? The optimization landscape of a ranking error function such as average precision is highly non-smooth and non-differentiable. It can be thought of as a discontinuous landscape with plateaus where a change in the parameters (and hence the RSV of certain documents) does not alter the final document ranking.

In order to directly optimize for a ranking function, Cohen et al. [1998] proposed a *pairwise approach* regarding document pairs as instances in the learning procedure and again viewing the learning problem as classification of document pairs. During the learning phase document pairs are generated from the rankings of the ground truth and for each pair a label (or score) is assigned indicating that the first document should be ranked higher or lower than the second. Based on this data a model is learned which is subsequently used to rank documents. Different learning models lead to different algorithms such as RankBoost [Freund et al., 2003] (boosting), Ranking SVM [Herbrich et al., 2000] (support vector machines) and RankNet [Burges et al., 2005] (neural networks) (cf. [Cao et al., 2007]). In the last case, Burges et al. [2005] proposed a straightforward modification of the backpropagation algorithm to learning ranking functions. RankNet is based on a two-layer NN, the input units represent the joint feature mapping of a query and a document $\phi(q, d)$ and the single output unit represents the relevance score that is subsequently used for ranking. In network training, for each document pair $(\phi(q, d_i), \phi(q, d_j))$, where d_i should be ranked higher than d_j, two forward propagations are performed, one for $\phi(q, d_i)$ and one for $\phi(q, d_j)$. The difference of the two activation values of the output unit is propagated back to compute the gradients with respect to the network parameters. In its official blog[1] Microsoft Live Search announced that Live Search employs the RankNet algorithm to enhance their ranking.

However, the pairwise approach comes with a certain limitation, since it reduces ranking to classification of document pairs. Hence, it assumes that the document pairs are independent of each other (i.e. the i.i.d. assumption underlying the above learning models). IR evaluation measures, however, share some unique properties. Most prominent, they favor true positives in the top position of the ranking. Recently, Cao et al. [2007] proposed a third approach that directly operates on ranked lists. This *listwise approach* takes (ranked) lists as learning instances and optimizes a loss function defined on a pair of lists, the predicted ranking and the ground truth. Experiments show that listwise approaches in general outperform both pointwise and pairwise approaches [Xu et al., 2008; Yue et al., 2007; Burges et al., 2006]. Another advantage of the listwise approach is that it is possible to take inter-document dependencies into account. Inter-document dependencies are necessary to model aspects such as topic-diversity of search results which themselves influence subjective relevance [Yue and Joachims, 2008].

Implicit Relevance Feedback The major problem inherent to all parameter learning approaches in IR[2] is the availability of sufficient training data (i.e. the lack thereof). In the context of IR and relevance feedback the reasons for this data acquisition bottleneck are two-fold:

- The users are reluctant to provide explicit feedback.
- The more complex a model is, that is the more parameter it has, the more labelled data is needed to estimate those parameters accurately.

[1] `http://blogs.msdn.com/livesearch/archive/2005/06/21/431288.aspx` (last visited 31.10.2008)

[2] Actually, it can be considered as the problem inherent to all supervised learning methods.

Recent studies show that implicit feedback is a valuable source to overcome this data acquisition bottleneck [Joachims and Radlinski, 2007]. For example, it is relatively easy for high-volume systems, such as web search engines, to collect user-interaction data such as click-through information (cf. [Manning et al., 2008]. This data, certainly not as reliable as explicit feedback but available in large quantities, turns out to be a valueable resource. However, a particular subtlety of certain types of implicit relevance feedback data such as click-through is the fact that it cannot be regarded as absolute but rather as relative feedback. Radlinski and Joachims [2005] show that, in the context of web search, the fact that a user skipped the first result in the ranking and immediately clicked on the second is an accurate indicator that the second web page is more relevant than the first. Whereas established relevance feedback techniques such as presented in Chapter 3 cannot handle this type of *preference* data, it is exactly what the pairwise approach as outlined above requires to learn a ranking (preference) function. However, it has to be noted that there is a considerable presentation bias associated with implicit relevance feedback indicators. For example, Radlinski and Joachims [2005] anticipate that users examine the ranking from top to bottom, which they empirically validated for their prototype using eye tracking. Their findings, however, do not generalize to different user interfaces. Furthermore, they assume that the generated document summaries (snippets) that are shown in the result ranking are indicative of the relevance of these documents (cf. [Manning et al., 2008, p.172]).

6.2.2 Ideas for Future Work

There are a number of issues and ideas which, the author thinks, deserve further investigation. The most urgent and worthwhile among these are:

- Validate the findings on different (larger) test collections. IR models should be evaluated on different collections in order to report reliable results.
- Contrast the evaluation results of the proposed model with an automatic controlled indexing approach based on automatic text categorization techniques [Sebastiani, 2002]. For example train a multi-class text classifier on the relevance feedback data, treating a document as a bag-of-words and the corresponding query concepts as the class labels (multi-class classification). The trained model is subsequently used to determine the concept to document connection weights based on the confidence of the classifier that the given document is labelled with the given concept.
- The fact that the APOSDLE client application has access to the operating system can be exploited to derive reliable implicit feedback indicators from the user behavior (e.g. reading time). This data might be used to build larger, more realistic and, first and foremost, cost-effective test collections (cf. [Radlinski et al., 2008]).

Chapter 7

Appendix A

7.1 Query Statistics

Query-ID	not-judged	non-rel	num-rel	concept
dme1	978	10	29	http://www.eads.net/UCM.owl#ART-SCENE
dme2	964	19	34	http://www.eads.net/UCM.owl#Action
dme3	969	20	28	http://www.eads.net/UCM.owl#Actor
dme4	951	53	13	http://www.eads.net/UCM.owl#Event
dme5	976	32	9	http://www.eads.net/UCM.owl#Object
dme6	909	16	92	http://www.eads.net/UCM.owl#Requirement
dme7	963	22	32	http://www.eads.net/UCM.owl#Scenario
dme9	979	17	21	http://www.eads.net/UCM.owl#UML_Use_Case_Diagram
dme10	888	19	110	http://www.eads.net/UCM.owl#Use_Case
dme11	916	83	18	http://www.eads.net/UCM.owl#Use_Case_Model
dme13	958	43	16	http://www.eads.net/owl-ontologies.com/human_activity.owl#Activity_Description
dme35	963	32	22	http://www.know-center.at/owl-ontologies/system_goals.owl#Eye_Star_Notation
dme36	987	13	17	http://www.know-center.at/owl-ontologies/system_goals.owl#Goal
dme39	966	38	13	http://www.know-center.at/owl-ontologies/system_goals.owl#SD_Model
dme41	963	38	16	http://www.know-center.at/owl-ontologies/system_goals.owl#SR_Model
dme42	1001	9	7	http://www.know-center.at/owl-ontologies/system_goals.owl#Soft_Goal
dme45	982	4	31	http://www.owl-ontologies.com/Top-Rescue.owl#ACRE_Methods
dme46	984	22	11	http://www.owl-ontologies.com/Top-Rescue.owl#Brainstorming
dme48	984	15	18	http://www.owl-ontologies.com/Top-Rescue.owl#Ethnography
dme49	984	19	14	http://www.owl-ontologies.com/Top-Rescue.owl#Interview
dme50	975	9	33	http://www.owl-ontologies.com/Top-Rescue.owl#Model
dme51	982	16	19	http://www.owl-ontologies.com/Top-Rescue.owl#Notation
dme52	984	14	19	http://www.owl-ontologies.com/Top-Rescue.owl#Observation
dme53	973	15	29	http://www.owl-ontologies.com/Top-Rescue.owl#RESCUE_Method
dme60	979	8	30	http://www.owl-ontologies.com/Top-Rescue.owl#Tool
dme63	973	28	16	http://www.owl-ontologies.com/Top-level_domain.owl#Project

Table 7.1: Query statistics showing the number of not-judged, non-relevant and relevant documents w.r.t. each query as well as the concept from the domain ontology referring to the query identifier.

Query	Num-Rel	Baseline				Backprop $\hat{C}$-D				Backprop				Backprop C-$\hat{C}$-D			
		infAP	P5	P10	P20	infAP	P5	P10	P20	infAP	P5	P10	P20	infAP	P5	P10	P20
all	697	0.7609	0.8769	0.7885	0.6846	0.7810	0.8923	0.8308	0.7058	0.6933	0.6615	0.61 54	0.5308	0.7974	0.9000	0.8385	0.7038
dme1	29	0.9269	1.0000	1.0000	0.9000	0.9269	1.0000	1.0000	0.9000	1.0000	1.0000	1.00 00	1.0000	0.9269	1.0000	1.0000	0.9000
dme10	110	0.9396	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9938	0.4000	0. 7000	0.7000	1.0000	1.0000	1.0000	1.0000
dme11	18	0.7854	1.0000	0.9000	0.6000	0.7854	1.0000	0.9000	0.6000	0.8366	1.0000	1.0 000	0.6000	0.7854	1.0000	0.9000	0.6000
dme13	16	0.6711	0.8000	0.6000	0.6000	0.6711	0.8000	0.6000	0.6000	0.3204	0.2000	0.1 000	0.0500	0.6711	0.8000	0.6000	0.6000
dme2	34	0.8328	1.0000	0.8000	0.8500	0.9094	1.0000	1.0000	0.9500	0.8696	0.8000	0.80 00	0.9000	0.9094	1.0000	1.0000	0.9500
dme3	28	0.6078	0.8000	0.8000	0.5500	0.7065	1.0000	0.9000	0.7000	0.6004	0.4000	0.60 00	0.7000	0.7065	1.0000	0.9000	0.7000
dme35	22	0.3595	0.8000	0.7000	0.5000	0.3595	0.8000	0.7000	0.5000	0.4043	1.0000	0.8 000	0.5000	0.3595	0.8000	0.7000	0.5000
dme36	17	0.4456	0.8000	0.6000	0.5000	0.4456	0.8000	0.6000	0.5000	0.5677	1.0000	0.8 000	0.5000	0.4456	0.8000	0.6000	0.5000
dme39	13	0.9589	1.0000	0.9000	0.6500	1.0000	1.0000	1.0000	0.6500	0.5258	1.0000	0.5 000	0.2500	1.0000	1.0000	1.0000	0.6500
dme4	13	0.1813	0.2000	0.1000	0.1500	0.2227	0.2000	0.3000	0.1500	0.1045	0.0000	0.00 00	0.1500	0.0839	0.0000	0.0000	0.1000
dme41	16	0.9228	1.0000	1.0000	0.7000	0.9228	1.0000	1.0000	0.7000	0.2102	0.0000	0.0 000	0.1500	0.9228	1.0000	1.0000	0.7000
dme42	7	0.8201	0.8000	0.6000	0.3500	0.8201	0.8000	0.6000	0.3500	0.9480	1.0000	0.60 00	0.3500	0.8201	0.8000	0.6000	0.3500
dme45	31	0.9936	1.0000	1.0000	1.0000	0.9936	1.0000	1.0000	1.0000	0.9935	1.0000	1.0 000	0.9000	0.9938	1.0000	1.0000	1.0000
dme46	11	0.5948	0.6000	0.6000	0.5500	0.5948	0.6000	0.6000	0.5500	0.6879	0.6000	0.7 000	0.5500	0.8355	0.8000	0.8000	0.5500
dme48	18	0.9228	1.0000	1.0000	0.8000	0.9228	1.0000	1.0000	0.8000	0.9145	1.0000	1.0 000	0.7500	0.9253	1.0000	1.0000	0.7500
dme49	14	0.7894	1.0000	0.8000	0.5500	0.7894	1.0000	0.8000	0.5500	0.6805	0.8000	0.8 000	0.6000	0.8643	1.0000	0.9000	0.5500
dme5	9	0.3535	0.6000	0.3000	0.1500	0.3535	0.6000	0.3000	0.1500	0.2583	0.4000	0.200 0	0.1000	0.3535	0.6000	0.3000	0.1500
dme50	33	0.9991	1.0000	1.0000	1.0000	0.9895	1.0000	1.0000	1.0000	0.7941	0.0000	0.0 000	0.2000	0.9934	1.0000	1.0000	1.0000
dme51	19	0.7174	0.8000	0.6000	0.7500	0.6872	0.8000	0.7000	0.7500	0.6093	0.8000	0.4 000	0.2500	0.8780	1.0000	0.9000	0.7500
dme52	19	0.9241	1.0000	1.0000	0.8500	0.9241	1.0000	1.0000	0.8500	0.9346	1.0000	1.0 000	0.8000	0.9765	1.0000	1.0000	0.9000
dme53	29	0.8239	1.0000	0.9000	0.8500	0.8224	1.0000	0.9000	0.8500	0.5583	0.2000	0.2 000	0.6000	0.8224	1.0000	0.9000	0.8500
dme6	92	0.9974	1.0000	1.0000	1.0000	0.9998	1.0000	1.0000	1.0000	0.9864	0.4000	0.40 00	0.2500	0.9998	1.0000	1.0000	1.0000
dme60	30	0.9254	1.0000	1.0000	0.8500	0.9254	1.0000	1.0000	0.8500	0.9717	1.0000	1.0 000	0.9500	0.9254	1.0000	1.0000	0.8500
dme63	16	0.6799	0.8000	0.7000	0.5500	0.6769	0.8000	0.7000	0.5500	0.7701	1.0000	1.0 000	0.5000	0.6769	0.8000	0.7000	0.5500
dme7	32	0.6981	1.0000	0.7000	0.6500	0.8797	1.0000	1.0000	0.9000	0.9429	1.0000	1.00 00	1.0000	0.8797	1.0000	1.0000	0.9000
dme9	21	0.9111	0.8000	0.9000	0.9000	0.9762	1.0000	1.0000	0.9500	0.5429	0.2000	0.40 00	0.5000	0.9765	1.0000	1.0000	0.9500

Table 7.2: Evaluation scores of the three runs and the baseline on query-level. The first column indicates the query identifier, the second shows the number of documents judged relevant w.r.t. the query. The evaluation measures are defined in Section 2.2. The first row shows the scores accumulated (num-rel) or averaged (evaluation measures) over all 26 queries.

7.2 Sequence Diagrams

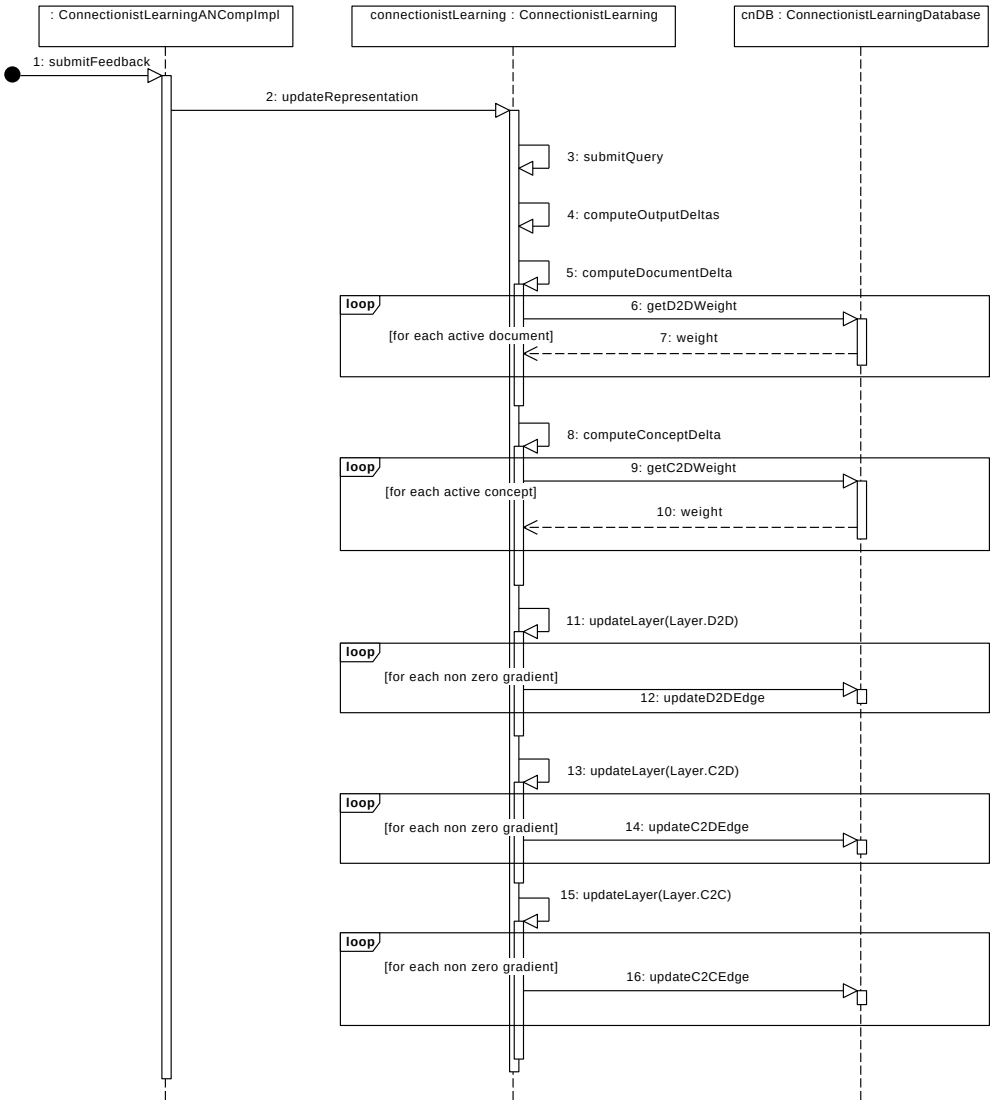

Figure 7.1: An UML sequence diagram showing the iteraction among the classes during an invocation of the `updateRepresentation` method of the `BackpropLearning` class. The interaction during the `submitQuery` method is modelled in the sequence diagram in Figure 5.7.

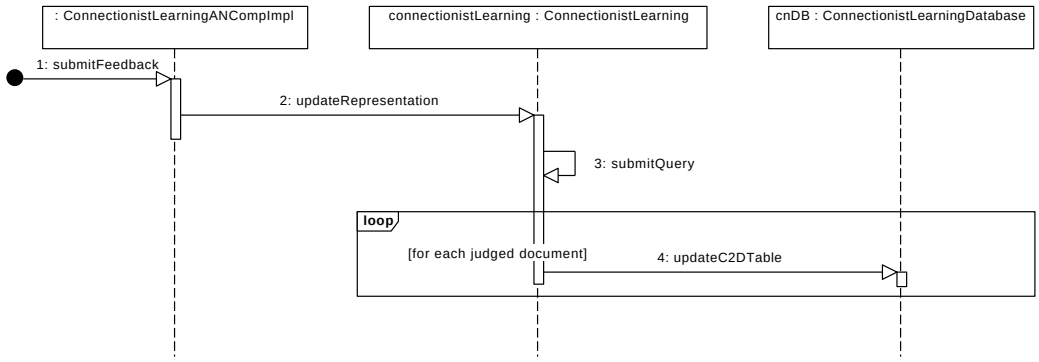

Figure 7.2: An UML sequence diagram showing the iteraction among the classes during an invocation of the `updateRepresentation` method of the `PerceptronLearning` class. The interaction during the `submitQuery` method is modelled in the sequence diagram in Figure 5.7.

7.3 Spring Configuration File

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xmlns:util="http://www.springframework.org/schema/util"
5         xsi:schemaLocation="
6   http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/
      spring-beans-2.0.xsd
7   http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring
      -util-2.0.xsd">
8     <import resource="../../../common/conf/CommonContext.xml" />
9
10    <bean id="ANLogger"
11       class="org.aposdle.associativenetwork.logging.ANLogger">
12       <constructor-arg index="0"
13          value="../associativenetwork/conf/AN.log4j.properties" />
14    </bean>
15
16    <bean id="ConnectionistLearningDatabase"
17       class="org.aposdle.associativenetwork.db.ConnectionistLearningDatabaseCache">
18       <property name="connectionistLearningDatabase">
19          <ref bean="ConnectionistLearningDatabaseReal" />
20       </property>
21
22    </bean>
23
24    <bean id="ConnectionistLearningDatabaseReal"
25       class="org.aposdle.associativenetwork.db.ConnectionistLearningDatabaseImpl">
26       <property name="associativeNetworkDatabase">
27          <ref bean="AssociativeNetworkDatabase" />
28       </property>
29       <property name="anDataSource">
30          <ref bean="ANDataSource" />
31       </property>
32
33    </bean>
34
35    <bean id="AssociativeNetworkDatabase"
36       class="org.aposdle.associativenetwork.db.AssociativeNetworkDatabaseCache">
37       <property name="associativeNetworkDatabase">
38          <ref bean="AssociativeNetworkDatabaseReal" />
39       </property>
```

```
40    </bean>
41
42    <bean id="AssociativeNetworkDatabaseReal"
43       class="org.aposdle.associativenetwork.db.AssociativeNetworkDatabaseImpl">
44      <property name="ANLogger">
45        <ref bean="ANLogger" />
46      </property>
47        <property name="similarConceptLowerBound_" value="0.7" />
48        <property name="anDataSource">
49        <ref bean="ANDataSource" />
50      </property>
51    </bean>
52
53    <bean id="ConnectionistLearningANService"
54       class="org.aposdle.associativenetwork.ConnectionistLearningANComponentImpl">
55      <property name="associativeNetworkDatabase">
56        <ref bean="ConnectionistLearningDatabase" />
57      </property>
58      <property name="conceptToDocumentLayer">
59        <ref bean="ConceptToDocumentLayer" />
60      </property>
61
62      <property name="conceptToConceptLayer">
63        <ref bean="ConceptToConceptLayer" />
64      </property>
65
66      <property name="documentToDocumentLayer">
67        <ref bean="DocumentToDocumentLayer" />
68      </property>
69
70      <property name="connectionistLearning">
71        <ref bean="BackpropConnectionistLearning" />
72      </property>
73
74    </bean>
75
76    <bean id="URIToKnowledgeArtefactTranformer"
77       class="org.aposdle.associativenetwork.topology.DummyUriToKnowledgeArtefactTransformer">
78    </bean>
79
80    <bean id="ConceptToDocumentLayer"
81       class="org.aposdle.associativenetwork.topology.ConceptToDocumentLayerImpl">
82      <constructor-arg>
83        <ref bean="ConnectionistLearningDatabase" />
84      </constructor-arg>
85      <property name="uRIToKnowledgeArtefactTranformer">
86        <ref bean="URIToKnowledgeArtefactTranformer" />
87      </property>
88      <property name="documentNodeFactory">
89        <ref bean="c2dlinearDocumentNodeFactory" />
90      </property>
91    </bean>
92
93    <bean id="ConceptToConceptLayer"
94       class="org.aposdle.associativenetwork.topology.ConceptToConceptLayerL1Norm">
95      <constructor-arg>
96        <ref bean="ConnectionistLearningDatabase" />
97      </constructor-arg>
98    </bean>
99
100   <bean id="DocumentToDocumentLayer"
101      class="org.aposdle.associativenetwork.topology.DocumentToDocumentLayerL1Norm">
102     <constructor-arg>
103       <ref bean="ConnectionistLearningDatabase" />
104     </constructor-arg>
105     <property name="uRIToKnowledgeArtefactTranformer">
106       <ref bean="URIToKnowledgeArtefactTranformer" />
107     </property>
108     <property name="documentNodeFactory">
109       <ref bean="d2dsigmoidDocumentNodeFactory" />
110     </property>
111   </bean>
```

```
112
113     <bean id="d2dsigmoidDocumentNodeFactory"
114       class="org.aposdle.associativenetwork.topology.DocumentNodeFactory">
115       <property name="activationFunction">
116         <ref bean="sigmoidActivationFunction" />
117       </property>
118     </bean>
119
120     <bean id="c2dlinearDocumentNodeFactory"
121       class="org.aposdle.associativenetwork.topology.DocumentNodeFactory">
122       <property name="activationFunction">
123         <ref bean="linearActivationFunction" />
124       </property>
125     </bean>
126
127     <bean id="linearActivationFunction"
128       class="org.aposdle.associativenetwork.topology.LinearActivationFunction">
129     </bean>
130     <bean id="sigmoidActivationFunction"
131       class="org.aposdle.associativenetwork.topology.SigmoidActivationFunction">
132     </bean>
133
134     <bean id="PerceptronConnectionistLearning"
135       class="org.aposdle.associativenetwork.learning.PerceptronConnectionistLearning">
136       <property name="connectionistLearningDatabase">
137         <ref bean="ConnectionistLearningDatabase" />
138       </property>
139       <property name="learningRate"
140         value="0.01f" />
141     </bean>
142
143     <bean id="BackpropConnectionistLearning"
144       class="org.aposdle.associativenetwork.learning.BackpropConnectionistLearning">
145       <property name="connectionistLearningDatabase">
146         <ref bean="ConnectionistLearningDatabase" />
147       </property>
148       <property name="learningRate"
149         value="0.01f" />
150     </bean>
151
152   </beans>
```

Listing 7.1: Spring XML configuration file. This file is used by the Spring dependency injection container to deploy software components and resolve object dependencies via setter injection.

Bibliography

James Allan [2007]. *Information Retrieval - Lecture Notes*. http://ciir.cs.umass.edu/cmpsci646/.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto [1999]. *Modern Information Retrieval*. Addison Wesley. ISBN 020139829X.

Brian T. Bartell [1994]. *Optimizing ranking functions: a connectionist approach to adaptive information retrieval*. PhD thesis, La Jolla, CA, USA.

R. K. Belew [1989]. *Adaptive information retrieval: using a connectionist representation to retrieve and learn about documents*. In *SIGIR '89: Proceedings of the 12th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–20. ACM Press, New York, NY, USA. ISSN 0163-5840. doi:http://dx.doi.org/10.1145/75334.75337.

Richard K. Belew [2001]. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge University Press. ISBN 0521630282.

Nicholas J. Belkin and Bruce B. Croft [1992]. *Information filtering and information retrieval: two sides of the same coin?* Commun. ACM, 35(12), pages 29–38. ISSN 0001-0782. doi:http://dx.doi.org/10.1145/138859.138861.

T. Berners-Lee, R. Fielding, and L. Masinter [2005]. *RFC 3986, Uniform Resource Identifier (URI): Generic Syntax*. http://tools.ietf.org/html/rfc3986.

Paul V. Biron and Donald H. Kraft [1995]. *New methods for relevance feedback: improving information retrieval performance*. In *SAC '95: Proceedings of the 1995 ACM symposium on Applied computing*, pages 482–487. ACM, New York, NY, USA. ISBN 0897916581. doi:http://dx.doi.org/10.1145/315891.316072.

Christopher M. Bishop [2006]. *Pattern Recognition and Machine Learning*. Springer. ISBN 0387310738.

Joshua Bloch [2008]. *Effective Java (2nd Edition) (The Java Series)*. Prentice Hall PTR, 2 Edition. ISBN 0321356683.

M. Boughanem, C. Chrisment, and C. Soule-Dupuy [1999]. *Query modification based on relevance back-propagation in an ad hoc environment*. Information Processing & Management, 35(2), pages 121–139. doi:http://dx.doi.org/10.1016/S0306-4573(99)00008-4.

Mohand Boughanem and Soulé C. Dupuy [1996]. *Mercure02: adhoc and routing tasks*. In *TREC*.

Mohand Boughanem, Karen Sauvagnat, and Cecile Laffaire [2003]. *Mercure at TREC 2003 Web track - Topic Distillation Task*. In *TREC*, pages 343–348.

T. Brauen [1969]. *Document vector modification*. In G. Salton (Editor), *The SMART Retrieval System*, chapter 24. Prentice-Hall, New Jersey.

Chris Buckley and Gerard Salton [1995]. *Optimization of relevance feedback weights*. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–357. ACM Press, New York, NY, USA. ISBN 0897917146. doi:http://dx.doi.org/10.1145/215206.215383.

Chris Buckley and Ellen M. Voorhees [2000]. *Evaluating evaluation measure stability*. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 33–40. ACM Press, New York, NY, USA. ISBN 1581132263. doi:http://dx.doi.org/10.1145/345508.345543.

Chris Buckley and Ellen M. Voorhees [2004]. *Retrieval evaluation with incomplete information*. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32. ACM Press, New York, NY, USA. ISBN 1581138814. doi:http://dx.doi.org/10.1145/1008992.1009000.

Chris Buckley, Gerard Salton, and James Allan [1994]. *The effect of adding relevance information in a relevance feedback environment*. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 292–300. Springer-Verlag New York, Inc., New York, NY, USA. ISBN 038719889X. doi:http://dx.doi.org/10.1145/160688.160689.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender [2005]. *Learning to rank using gradient descent*. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM Press, New York, NY, USA. ISBN 1595931805. doi:http://dx.doi.org/10.1145/1102351.1102363.

Christopher J. C. Burges, Robert Ragno, and Quoc V. Le [2006]. *Learning to Rank with Nonsmooth Cost Functions*. In Bernhard Schölkopf, John C. Platt, Thomas Hoffman, Bernhard Schölkopf, John C. Platt, and Thomas Hoffman (Editors), *NIPS*, pages 193–200. MIT Press. ISBN 0-262-19568-2.

Vannevar Bush [1945]. *As We May Think*. Atlantic Monthly.

Stefan Büttcher, Charles L. A. Clarke, and Ian Soboroff [2006]. *The TREC 2006 Terabyte Track*. In *15th Text REtrieval Conference (TREC 2006)*.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li [2007]. *Learning to rank: from pairwise approach to listwise approach*. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, New York, NY, USA. ISBN 9781595937933. doi:http://dx.doi.org/10.1145/1273496.1273513.

Claudio Carpineto, Renato de Mori, Giovanni Romano, and Brigitte Bigi [2001]. *An information-theoretic approach to automatic query expansion*. ACM Trans. Inf. Syst., 19(1), pages 1–27. ISSN 1046-8188. doi:http://dx.doi.org/10.1145/366836.366860.

Y. K. Chang, C. Cirillo, and J. Razon [1971]. *Evaluation of feedback retrieval using modified freezing, residual collection, and test and control groups*. In Gerald Salton (Editor), *The SMART retrieval system — experiments in automatic document processing*, pages 355–370.

Conny Christl, Chiara Ghidini, Joanna Guss, Stefanie Lindstaedt, Viktoria Pammer, Marco Rospocher, Peter Scheir, and Luciano Serafini [2008]. *Deploying Semantic Web Technologies for Work Integrated Learning in Industry - A Comparison: SME vs. Large Sized Company*. pages 709–722. doi:http://dx.doi.org/10.1007/978-3-540-88564-1_45.

Cyril W. Cleverdon [1991]. *The Significance of the Cranfield Tests on Index Languages*. In Abraham Bookstein, Yves Chiaramella, Gerard Salton, Vijay V. Raghavan, Abraham Bookstein, Yves Chiaramella, Gerard Salton, and Vijay V. Raghavan (Editors), *SIGIR*, pages 3–12. ACM. ISBN 0-89791-448-1.

William W. Cohen, Robert E. Schapire, and Yoram Singer [1998]. *Learning to Order Things*. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla (Editors), *Advances in Neural Information Processing Systems*, volume 10. The MIT Press.

William S. Cooper, Fredric C. Gey, and Daniel P. Dabney [1992]. *Probabilistic retrieval based on staged logistic regression*. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 198–210. ACM, New York, NY, USA. ISBN 0-89791-523-2. doi:http://dx.doi.org/10.1145/133160.133199.

E. Cortez, Sang C. Park, and Seonghee Kim [1995]. *The hybrid application of an inductive learning method and a neural network for intelligent information retrieval*. Information Processing & Management, 31(6), pages 789–813. ISSN 03064573. doi:http://dx.doi.org/10.1016/0306-4573(95)00015-9.

F. Crestani [1997]. *Application of Spreading Activation Techniques in Information Retrieval*. Artificial Intelligence Review, V11(6), pages 453–482. doi:http://dx.doi.org/10.1023/A:1006569829653%0D%0A.

F. Crestani, M. Lalmas, Cornelis J. van Rijsbergen, and I. Campbell [1998]. *Is this document relevant?...Probably: A survey of probabilistic models in information retrieval*.

Fabio Crestani [1993]. *Learning Strategies for an Adaptive Information Retrieval System Using Neural Networks*. In *In Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 244–249.

Fabio Crestani and Cornelis J. van Rijsbergen [1997]. *A Model for Adaptive Information Retrieval*. Journal of Intelligent Information Systems, 8(1), pages 29–56. doi:http://dx.doi.org/10.1023/A:1008601616486.

Carolyn J. Crouch, Donald B. Crouch, and Krishnamohan Nareddy [1994]. *A connectionist model for information retrieval based on the vector space model*. Int. J. Expert Syst., 7(2), pages 139–163. ISSN 0894-9077.

S. Cunningham, G. Holmes, J. Littin, R. Beale, and I. Witten [1997]. *Applying connectionist models to information retrieval*.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman [1990]. *Indexing by Latent Semantic Analysis*. Journal of the American Society of Information Science, 41(6), pages 391–407.

Martin Fowler [2003]. *UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition*. Addison-Wesley Professional. ISBN 0321193687.

Martin Fowler [2004]. *Inversion of Control Containers and the Dependency Injection pattern*. `http://www.martinfowler.com/articles/injection.html`.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer [2003]. *An efficient boosting algorithm for combining preferences*. J. Mach. Learn. Res., 4, pages 933–969.

Norbert Fuhr [1989]. *Models for retrieval with probabilistic indexing*. Inf. Process. Manage., 25(1), pages 55–72. ISSN 0306-4573. doi:http://dx.doi.org/10.1016/0306-4573(89)90091-5.

Norbert Fuhr [1992]. *Probabilistic Models in Information Retrieval*. The Computer Journal, 35(3), pages 243–255.

Norbert Fuhr and Chris Buckley [1991]. *A probabilistic learning approach for document indexing*. ACM Trans. Inf. Syst., 9(3), pages 223–248. ISSN 1046-8188. doi:http://dx.doi.org/10.1145/125187.125189.

Norbert Fuhr and Ulrich Pfeifer [1994]. *Probabilistic information retrieval as a combination of abstraction, inductive learning, and probabilistic assumptions*. ACM Trans. Inf. Syst., 12(1), pages 92–115. ISSN 1046-8188. doi:http://dx.doi.org/10.1145/174608.174612.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides [1995]. *Design Patterns*. Addison-Wesley Professional. ISBN 0201633612.

Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom [2008]. *Database Systems: The Complete Book*. Pearson Education (US), 2International Ed Edition. ISBN 0131354280.

Donna Harman [1992]. *Relevance feedback revisited*. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 1–10. ACM, New York, NY, USA. ISBN 0897915232. doi:http://dx.doi.org/10.1145/133160.133167.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer [2000]. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA.

David Hull [1994]. *Improving text retrieval for the routing problem using latent semantic indexing*. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–291. Springer-Verlag New York, Inc., New York, NY, USA. ISBN 038719889X.

Raj D. Iyer, David D. Lewis, Robert E. Schapire, Yoram Singer, and Amit Singhal [2000]. *Boosting for document routing*. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 70–77. ACM, New York, NY, USA. ISBN 1581133200. doi:http://dx.doi.org/10.1145/354756.354794.

Thorsten Joachims [1997]. *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization*. In Douglas H. Fisher (Editor), *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151. Morgan Kaufmann Publishers, San Francisco, US, Nashville, US.

Thorsten Joachims and Filip Radlinski [2007]. *Search Engines that Learn from Implicit Feedback*. Computer, 40(8), pages 34–40. ISSN 0018-9162. doi:http://dx.doi.org/10.1109/MC.2007.289.

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay [2005]. *Accurately interpreting clickthrough data as implicit feedback*. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161. ACM Press, New York, NY, USA. ISBN 1595930345. doi:http://dx.doi.org/10.1145/1076034.1076063.

H. Joho, J. Urban, R. Villa, J. M. Jose, and C. J. van Rijsbergen [2008]. *AIR 2006: First international workshop on adaptive information retrieval*. SIGIR Forum, 42(1), pages 63–66.

Eric Jones, Travis Oliphant, Pearu Peterson, and Others [2001]. *SciPy: Open source scientific tools for Python*.

J. Jose, H. Joho, and C. van Rijsbergen [2008]. *Adaptive information retrieval: Introduction to the special topic issue of information processing and management*. Information Processing & Management, 44(6), pages 1819–1821. ISSN 03064573. doi:http://dx.doi.org/10.1016/j.ipm.2008.08.002.

Jon Kleinberg and Éva Tardos [2005]. *Algorithm Design*. Addison Wesley. ISBN 0321295358.

Bart Kosko [1988]. *Bidirectional associative memories*. IEEE Trans. Syst. Man Cybern., 18(1), pages 49–60. ISSN 0018-9472. doi:http://dx.doi.org/10.1109/21.87054.

K. L. Kwok [1989]. *A Neural Network for Probabilistic Information Retrieval*. In Nicholas J. Belkin, C. J. van Rijsbergen, Nicholas J. Belkin, and C. J. van Rijsbergen (Editors), *SIGIR*, pages 21–30. ACM. ISBN 0-89791-321-3.

K. L. Kwok [1995]. *A network approach to probabilistic information retrieval*. ACM Trans. Inf. Syst., 13(3), pages 324–353. ISSN 1046-8188. doi:http://dx.doi.org/10.1145/203052.203067.

David D. Lewis and Karen S. Jones [1996]. *Natural Language Processing for Information Retrieval*. Communications of the ACM, 39(1), pages 92–101.

Stefanie Lindstaedt and Harald Mayer [2006]. *A Storyboard of the APOSDLE Vision*. pages 628–633. doi:http://dx.doi.org/10.1007/11876663_64.

David J. C. Mackay [2002]. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press. ISBN 0521642981.

T. Mandl [2000]. *Tolerant Information Retrieval with Backpropagation Networks*. Neural Computing & Applications, 9(4), pages 280–289. doi:http://dx.doi.org/10.1007/s005210070005.

Thomas Mandl [2001]. *Tolerant Information Retrieval: Neural Networks for adaptivity and flexibility in searching*. PhD thesis, University of Hildesheim.

Christopher D. Manning and Hinrich Schtze [1999]. *Foundations of Statistical Natural Language Processing*. The MIT Press. ISBN 0262133601.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze [2008]. *Introduction to Information Retrieval*. Cambridge University Press. ISBN 0521865719.

M. E. Maron and J. L. Kuhns [1960]. *On Relevance, Probabilistic Indexing and Information Retrieval*. J. ACM, 7(3), pages 216–244. ISSN 0004-5411. doi:http://dx.doi.org/10.1145/321033.321035.

Stefano Mizzaro [1998]. *How many relevances in information retrieval?* Interacting with Computers, 10(3), pages 303–320.

Josiane Mothe [1994]. *Search mechanisms using neural network model, comparison with vector space model*. In *4th RIAO Intelligent Multimedia Information Retrieval Systems and Management*, pages 275–294.

Ramesh Nallapati [2004]. *Discriminative models for information retrieval*. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71. ACM, New York, NY, USA. ISBN 1581138814. doi:http://dx.doi.org/10.1145/1008992.1009006.

Judea Pearl [1988]. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann. ISBN 1558604790.

Jay M. Ponte and Bruce B. Croft [1998]. *A language modeling approach to information retrieval*. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM Press, New York, NY, USA. ISBN 1581130155. doi:http://dx.doi.org/10.1145/290941.291008.

Filip Radlinski and Thorsten Joachims [2005]. *Query chains: learning to rank from implicit feedback*. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248. ACM Press, New York, NY, USA. ISBN 159593135X. doi:http://dx.doi.org/10.1145/1081870.1081899.

Filip Radlinski, Madhu Kurup, and Thorsten Joachims [2008]. *How does clickthrough data reflect retrieval quality?* In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining*, pages 43–52. ACM, New York, NY, USA. ISBN 978-1-59593-991-3. doi:http://dx.doi.org/10.1145/1458082.1458092.

S. E. Robertson [1990]. *On term selection for query expansion*. J. Doc., 46(4), pages 359–364. ISSN 0022-0418.

S. E. Robertson [1997]. *The probability ranking principle in IR*. pages 281–286.

S. E. Robertson and Sparck K. Jones [1976]. *Relevance weighting of search terms*. Journal of the American Society for Information Science, 27(3), pages 129–146. doi:http://dx.doi.org/10.1002/asi.4630270302.

S. E. Robertson, S. Walker, and M. Beaulieu [2000]. *Experimentation as a way of life: Okapi at TREC*. Information Processing & Management, 36(1), pages 95–108. doi:http://dx.doi.org/10.1016/S0306-4573(99)00046-1.

Stephen E. Robertson, M. E. Maron, and William S. Cooper [1982]. *The Unified Probabilistic Model for IR*. In *SIGIR*, pages 108–117.

J. J. Rocchio [1971]. *Relevance Feedback in Information Retrieval*. Prentice Hall, Englewood, Cliffs, New Jersey.

David E. Rumelhart and James L. Mcclelland [1986]. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition : Foundations (Parallel Distributed Processing)*. MIT Press. ISBN 0262181207.

Stuart J. Russell and Peter Norvig [2002]. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall. ISBN 0137903952.

Ian Ruthven [2003]. *Re-examining the potential effectiveness of interactive query expansion*. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 213–220. ACM Press, New York, NY, USA. ISBN 1581136463. doi:http://dx.doi.org/10.1145/860435.860475.

Ian Ruthven and Mounia Lalmas [2003]. *A survey on the use of relevance feedback for information access systems*. Knowl. Eng. Rev., 18(2), pages 95–145. ISSN 0269-8889. doi:http://dx.doi.org/10.1017/S0269888903000638.

Gerard Salton [1989]. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0-201-12227-8.

Gerard Salton and Chris Buckley [1997]. *Improving retrieval performance by relevance feedback*. pages 355–364.

Mark Sanderson and Justin Zobel [2005]. *Information retrieval system evaluation: effort, sensitivity, and reliability*. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169. ACM Press, New York, NY, USA. ISBN 1595930345. doi:http://dx.doi.org/10.1145/1076034.1076064.

Tefko Saracevic [2007]. *Relevance: A review of the literature and a framework for thinking on the notion in information science. Part II: nature and manifestations of relevance*. Journal of the American Society of Information Sciences and Technologies, 58(13), pages 1915–1933. doi:http://dx.doi.org/http://dx.doi.org/10.1002/asi.v58:13.

Robert E. Schapire, Yoram Singer, and Amit Singhal [1998]. *Boosting and Rocchio applied to text filtering*. In Bruce W. Croft, Alistair Moffat, Cornelis J. van Rijsbergen, Ross Wilkinson, and Justin Zobel (Editors), *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 215–223. ACM Press, New York, US, Melbourne, AU.

Peter Scheir [2008]. *Associative Retrieval for the Semantic Web - A network-based search approach in consideration of semantic and content-based similarity*. PhD thesis.

Peter Scheir, Chiara Ghidini, and Stefanie N. Lindstaedt [2007a]. *Improving Search on the Semantic Desktop using Associative Retrieval Techniques*. In *I-MEDIA 2007 and I-SEMANTICS 2007*, pages 221–228.

Peter Scheir, Michael Granitzer, and Stefanie N. Lindstaedt [2007b]. *Evaluation of an Information Retrieval System for the Semantic Desktop using Standard Measures from Information Retrieval*. In Alexander Hinneburg (Editor), *LWA 2007: Lernen - Wissen - Adaption*, pages 269–272. Martin-Luther-University Halle-Wittenberg.

Hinrich Schütze, David A. Hull, and Jan O. Pedersen [1995]. *A Comparison of Classifiers and Document Representations for the Routing Problem*. In *Research and Development in Information Retrieval*, pages 229–237.

Fabrizio Sebastiani [2002]. *Machine learning in automated text categorization*. ACM Computing Surveys, 34(1), pages 1–47.

Robert Sedgewick [2003]. *Algorithms in Java, Part 5: Graph Algorithms (3rd Edition)*. Addison-Wesley Professional, 3 Edition. ISBN 0201361213.

Amit Singhal, Chris Buckley, and Mandar Mitra [1996]. *Pivoted document length normalization*. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29. ACM, New York, NY, USA. ISBN 0897917928. doi:http://dx.doi.org/10.1145/243199.243206.

Amit Singhal, Mandar Mitra, and Christopher Buckley [1997]. *Learning routing queries in a query zone*. In *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, pages 25–32. Philadelphia, US.

Ian Soboroff and Nick Craswell [2006]. *Overview of the TREC 2006 Enterprise Track*. In *In Proceedings of TREC-06*.

Ian Soboroff and Stephen Robertson [2003]. *Building a filtering test collection for TREC 2002*. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 243–250. ACM, New York, NY, USA. ISBN 1581136463. doi:http://dx.doi.org/10.1145/860435.860481.

Howard R. Turtle and Bruce W. Croft [1990]. *Inference Networks for Document Retrieval*. In Jean L. Vidick and Jean L. Vidick (Editors), *SIGIR*, pages 1–24. ACM. ISBN 0-89791-408-2.

C. J. van Rijsbergen [1979]. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow.

Ellen M. Voorhees [2008]. *On test collections for adaptive information retrieval*. Information Processing & Management, 44(6), pages 1879–1885. doi:http://dx.doi.org/10.1016/j.ipm.2007.12.011.

Ellen M. Voorhees and Chris Buckley [2002]. *The effect of topic set size on retrieval experiment error*. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 316–323. ACM, New York, NY, USA. ISBN 1581135610. doi:http://dx.doi.org/10.1145/564376.564432.

Ellen M. Voorhees and Donna Harman [1997]. *Overview of the Sixth Text REtrieval Conference (TREC-6)*. In *TREC*, pages 1–24.

Ellen M. Voorhees and Donna K. Harman [2005]. *TREC: Experiment and Evaluation in Information Retrieval*. Digital Libraries and Electronic Publishing. MIT Press. ISBN 0262220733.

Wolfgang Wahlster, Henry Lieberman, and James Hendler [2002]. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. The MIT Press. ISBN 0262062321.

Ryen W. White, Ian Ruthven, and Joemon M. Jose [2005]. *A study of factors affecting the utility of implicit relevance feedback*. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 35–42. ACM Press, New York, NY, USA. ISBN 1595930345. doi:http://dx.doi.org/10.1145/1076034.1076044.

Ryen W. White, Joemon M. Jose, and Ian Ruthven [2006]. *An implicit feedback approach for interactive information retrieval*. Inf. Process. Manage., 42(1), pages 166–190. ISSN 0306-4573. doi:http://dx.doi.org/10.1016/j.ipm.2004.08.010.

Ross Wilkinson and Philip Hingston [1991]. *Using the cosine measure in a neural network for document retrieval*. In *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 202–210. ACM, New York, NY, USA. ISBN 0897914481. doi:http://dx.doi.org/10.1145/122860.122880.

Ian H. Witten and Eibe Frank [2005]. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, Second Edition. ISBN 0120884070.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell [1999]. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann. ISBN 1558605703.

Wong and Y. Y. Yao [1988]. *Linear structure in information retrieval*. In *SIGIR '88: Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 219–232. ACM, New York, NY, USA. ISBN 2-7061-0309-4. doi:http://dx.doi.org/10.1145/62437.62452.

S. K. M. Wong and Y. Y. Yao [1989]. *A probability distribution model for information retrieval*. Inf. Process. Manage., 25(1), pages 39–53. ISSN 0306-4573. doi:http://dx.doi.org/10.1016/0306-4573(89)90090-3.

Zhibiao Wu and Martha Palmer [1994]. *Verb semantics and lexical selection*. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138. New Mexico State University, Las Cruces, New Mexico.

Jinxi Xu and Bruce W. Croft [1996]. *Query Expansion Using Local and Global Document Analysis*. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11.

Jinxi Xu and Bruce W. Croft [2000]. *Improving the effectiveness of information retrieval with local context analysis*. ACM Trans. Inf. Syst., 18(1), pages 79–112. ISSN 1046-8188. doi:http://dx.doi.org/10.1145/333135.333138.

Jun Xu, Tie Y. Liu, Min Lu, Hang Li, and Wei Y. Ma [2008]. *Directly optimizing evaluation measures in learning to rank*. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 107–114. ACM, New York, NY, USA. ISBN 978-1-60558-164-4. doi:http://dx.doi.org/10.1145/1390334.1390355.

Emine Yilmaz and Javed A. Aslam [2006]. *Estimating average precision with incomplete and imperfect judgments*. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 102–111. ACM, New York, NY, USA. ISBN 1595934332. doi:http://dx.doi.org/10.1145/1183614.1183633.

Yisong Yue and Thorsten Joachims [2008]. *Predicting diverse subsets using structural SVMs*. In William W. Cohen, Andrew Mccallum, and Sam T. Roweis (Editors), *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 1224–1231. ACM. ISBN 978-1-60558-205-4. doi:http://dx.doi.org/10.1145/1390156.1390310.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims [2007]. *A support vector method for optimizing average precision*. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278. ACM Press, New York, NY, USA. ISBN 9781595935977. doi:http://dx.doi.org/10.1145/1277741.1277790.