Chapter 5

Towards an adaptive IR system

5.1 Introduction

[1]Adaptive techniques are applicable to those computer systems that are subject to repeated use by particular users. The user's needs will be quite specific, whereas the system will be generally applicable. One such computer system is an online Information Retrieval System (IRS); a user can retrieve documents in any subject area through interactive sessions, and may use the system several times to satisfy a specific information need. The standard IRS does not, however, adapt to a user's need over repeated sessions: each time the system is used, the IRS starts from its general stance. Our work is aimed at overcoming this failing.

IRS vary in their types of input and output: a common but somewhat rigid type of input is for users to represent their information need in a binary form through boolean expressions. The output from these systems tends to be an unordered list of documents and the query formulation requires knowledge of boolean logic.

Systems accepting natural language input are more flexible when accepting user queries; although the mapping of the information need to the query may be easier for the user, interpreting and processing queries in this form is not as straightforward. Unlike boolean queries, the relationship between terms in the query is not explicit. A *probabilistic* IRS which accepts natural language input can rank the output documents by attaching scores (numerical measures) to the documents in the collection, based on the current

[1]This Chapter is slightly adapted from a paper to be given by Ayşe Göker and T L McCluskey at *ISMIS '91: International symposium on methodologies for intelligent systems*, Charlotte, North Carolina, Oct 16–19 1991.

query [9]. The order of the references output reflects their probability of relevance, where relevance can be seen as a relationship between the document and the need. To improve the performance of a Probabilistic IRS further, *relevance feedback* is sought from a user. This permits users to choose documents which are relevant to their need, and allows the system to expand the original query using additional terms from the relevant documents.

We take this type of system one step further and introduce a learning component which adapts to a user over a period of online sessions. Relevance feedback from these sessions can be input to an *incremental concept learner* [6] which learns a 'context' or subject area for that user, and can utilise it to influence the ordering or inclusion of documents output in later sessions. In this paper we will first introduce the necessary background to the probabilistic model, then specify a learning component for a probabilistic IRS based on input from relevance feedback, and finally describe some results from a prototype implementation of the learning component.

5.2 The probabilistic model for an IRS

When a query is input to an IRS, the words can be extracted and stemmed in order to produce *search terms*. Probabilistic models attach weights to search terms and apply a match function to rank individual documents. The value or usefulness of a search term is not an uncommon concern in information retrieval. Relevance weighting theory [11] is an approach to quantify the value of a term based on its performance within a database: assuming document terms are independent of each other, the theory argues that the following formula can be used to achieve optimum performance. The weight of a term given by this formula is used in calculating individual document scores:

$$w_t \;=\; log\left(\frac{p/(1-p)}{q/(1-q)}\right) \tag{5.1}$$

where w_t = the weight of the term
p = the probability that the term will occur in relevant documents
q = the probability that the term will occur in non-relevant documents.

This formula uses the relevance feedback information obtained from the user. Prior to relevance feedback, in the first iteration of processing the user's query, the formula reduces to collection frequency weighting [1]. With this weighting rare terms are given high weight and the more frequently occurring terms are given low weights. After relevance feedback with p and q estimated, however, the term weight can be expressed as follows (the derivation of which is given in [11]:

$$w_t \;=\; log\left(\frac{(r+0.5)\;/\;(R-r+0.5)}{(n-r+0.5)\;/\;(N-n-R+r+0.5)}\right) \tag{5.2}$$

where w_t = the weight of the term
n = the number of postings for the term
(number of documents containing the term)
N = a constant larger than n for the most highly posted term in the search
R = number of records chosen as relevant
r = number of chosen records containing the term

Thus, the relevance feedback from the user enables a form of query expansion. Equation 5.2 can be used to select new terms and to calculate weights for them in order to merge them with the original query. The matching value of each document retrieved for that query is the sum of the weights of the query terms that index it.

Recently, a theoretical argument for improving this baseline formula was presented in [10]. Here, a_t is used as a measure for term selection instead of w_t. This improved version is given as

$$a_t = w_t(p - q) \tag{5.3}$$

Later we shall comment on how the use of this improved equation compares with output from the learning component explained in the next section.

5.3 The learning component

5.3.1 Inputs to the learner

To describe the learning mechanism, we will introduce the simplifying assumption that a document record (d) contains a unique identifier (d_{id}) together with a set of stemmed terms which make up the record (d_{st}). We also assume that each stemmed search term is stored in a structure (m) containing the structure's identifier (m_{id}, the stemmed search term), the weight of the term (m_{wt}), the number of relevant documents that contain the term (m_{dc}), and a record of the origin of the term. We say that a term m *covers* a document d if $m_{id} \in d_{st}$.

Relevance feedback results in the highest weighted search terms being collected from the relevant document records (32 in the the Okapi system described in section 5.4). Hence each *learning session* begins with input to the learning component, made up from:

- A set of document records D which have been picked as relevant by the user.
- A set of term structures M containing those terms (from D) with the highest weight, using the probabilistic model described in section 5.2.

To obtain maximum benefit from relevance feedback, a pre-condition of the use of the learning component is that the user is conducting a series of searches within a common context (there is evidence that users tend to repeat searches or conduct a series of closely related searches, hence this condition is not unrealistic). Within this framework it is assumed that a document marked as relevant in one session, while not necessarily still regarded as relevant in a later session, nevertheless remains representative of the context for the later search. Using relevance feedback from each of the searches the system will then create and evolve a *context* C, representing the users information need, that can be used in future searches to influence the ordering of documents displayed.

5.3.2 Formulating and evolving the context

The context C must be acquired and kept in an operational form, so that it may be easily utilised in future searches. Hence we define C to be a set of unique term structures, each with a corresponding set of attributes which adjust incrementally over a period of learning sessions.

C should also be representative of all the examples provided (i.e. complete). This is accomplished by building up a set of terms which cover all the relevant documents during each learning session. Operationality is supported by admitting to C only a *minimal* cover of the set of relevant documents. For the purposes of brevity, in this paper we will assume that a context term has attributes similar to the term structures, and in the specification below, we use the convention that X_{id}, where X is a set, stands for the set of identifiers of X's elements.

An IRS online session may result in zero, one or more learning sessions, depending on how many times relevance feedback was provided; C incrementally evolves after each learning session, as follows:

- Given inputs M and D, first remove any of the relevant documents which have been input in a previous learning session, since these documents will already be covered by the current context description. Hence D may shrink and the terms in M may have to have their attributes adjusted accordingly. In the absence of some of the documents a term's m_{dc} component may be reduced to zero in which case the whole term structure is deleted from the input.

- Next, the system forms a complete, minimally sized covering of the resulting documents in D in the form of a subset of M. It uses two bias criteria: favouring *terms which cover more than one relevant document* and *terms of high weight*. The covering takes the form of a subset of M we call A, standing for Active set. All unused term structures, that is the set $M - A$, are put into a Passive set called P. A itself is the disjoint union of two component subsets A' and A''. A' contains all those terms

which cover more than one document, and are not redundant in the sense that the documents they cover are not all covered by another term:

$$A' = \{m \in M | m_{dc} > 1 \wedge (\forall n \in M : n_{dc} > 1 \Rightarrow \exists d \in D : (n_{id} \notin d_{st} \wedge m_{id} \in d_{st}) \vee m = n)\}$$

A'' completes the set of terms which cover D, picking the terms of highest weight from the documents that were left out of the covering provided by A':

$$A'' = \{m \in M | \exists d \in D : (m_{id} \in d_{st} \wedge d_{st} \cap A'_{id} = \emptyset \wedge \forall x \in M : (x_{id} \in d_{st} \Rightarrow m_{wt} \geq x_{wt}))\}$$

Thus the set A is representative of all the relevant documents from the training session. Note that A'' may well be empty, if A' provides a full covering of D.

- If no previous learning has taken place then A is taken to be our initial context C. On the other hand, the system merges A into the existing set of terms which formed the old context. We will describe this by calling the former context $OldC$, and the new context $NewC$. Likewise we have the old passive set of terms $OldP$ and the new set $NewP$.

  $NewC$ is the disjoint union of three sets:

  - context terms unaffected by the new training examples:
    $X = \{c | c \in OldC \wedge c_{id} \notin (A \cup P)_{id}\}$
  - incremented context term structures whose names appear in both the new and the old data:
    $Y = \{add(c, m) | m_{id} = c_{id} \wedge m_{id} \in (A \cup P)_{id} \wedge c_{id} \in (OldC \cup OldP)_{id}\}$
  - new terms that appear in A:
    $Z = \{m | m \in A \wedge m_{id} \notin (OldC \cup OldP)_{id}\}$

Here *add* is a function which takes two term structures which share the same identifier, and produces a new term structure with attributes combined. Each term's weight will be added together, as will be their document count. Thus $NewC$ will be the set $X \cup Y \cup Z$. Finally, $NewP$ is formed by adding all those members of P to $OldP$ which were not already present in $OldC$ or $OldP$:

$$NewP = \{m | m \in P \wedge m_{id} \notin (OldC \cup OldP)_{id}\}$$

5.4 Testing context formulation

5.4.1 The Okapi system

The learning component described derives its input from online session logs from the use of Okapi. In this section we will briefly describe Okapi's operation, and show some initial results from a prototype implementation of the learning component.

Okapi is based on the probabilistic model, and uses relevance feedback to improve its effectiveness, as described in 5.2. The system is illustrated in Appendix A. It has three databases available in the current implementation: INSPEC - computer science and information technology sub-section; LISA - Library and Information Science Abstracts; and the City University Library Catalogue. Academic users have access through the university's local area network and from dedicated terminals situated in various locations such as the university library.

After selecting a database, users type in their queries in the form of natural language. Some input preprocessing such as parsing and stemming is performed in order to derive a set of terms that represent the query. Weights are calculated for these terms and the documents are ranked accordingly. Brief details of the documents (title, author etc) are then shown to the user in a document list. This is displayed in descending weight order, aiming to reflect the probability of the document's relevance. If a user chooses to see more details of documents from the list, these are displayed and he/she is prompted to make a relevance judgement on each one. Relevance feedback is then applied using the baseline formula (equation 5.2) and if the user chooses to do so, the initial query may be expanded using stemmed terms from the documents deemed relevant, and a further document list is displayed to the user on this basis [17].

Analysis of user's logs shows that their queries consist of two to three words on average and it is not uncommon for this description of the user's information need to be too general (4.4.2). This leads to a major problem: the resulting document list displayed to the user may consist of screenfulls of documents grouped in the same relevance band. When queries are too general or ambiguous, these long *weight-blocks*, may contain many irrelevant documents and can frustrate the user.

5.4.2 An example

Knowledge of a user's needs from earlier sessions would help improve the document ordering within the weight-blocks so as to better suit his/her need, and is precisely the kind of problem we expect an adapting IRS to overcome. To illustrate this, and the workings of the learner, we present an example using three separate online sessions, each involving one query and one set of

Table 5.1:

Query	Active terms	Passive terms
q(1)	{c(directed,86,4,[m(1)]), c(edit,69,2,[m(1)]), c(graf,86,5,[m(1)]), c(grammar,105,5,[q(1),m(1)]), c(rewrit,78,2,[m(1)]), c(syntax,72,2,[m(1)])}	18 passive terms
q(2)	{c(directed,86,4,[m(1)]), c(edit,69,2,[m(1)]), c(graf,86,5,[m(1)]), c(grammar,105,5,[q(1),m(1)]), c(rewrit,78,2,[m(1)]), c(syntax,72,2,[m(1)]), c(concurrent,107,9,[q(2),m(2)]), c(grafic,84,9,[m(2)]), c(interprocess,83,2,[m(2)]), c(notation,72,3,[m(2)]), c(program,63,9,[m(2)]), c(visualisation,61,2,[m(2)])}	36 passive terms
q(3)	{c(concurrent,107,9,[q(2),m(2)]), c(directed,86,4,[m(1)]), c(edit,69,2,[m(1)]), c(graf,86,5,[m(1)]), c(grafic,84,9,[m(2)]), c(grammar,105,5,[q(1),m(1)]), c(interprocess,83,2,[m(2)]), c(notation,72,3,[m(2)]), c(program,63,9,[m(2)]), c(rewrit,78,2,[m(1)]), c(syntax,72,2,[m(1)]), c(visualisation,61,2,[m(2)]), c(parallel,73,6,[q(3),m(3)]), c(supercomputer,72,3,[m(3)]), c(tool,71,6,[m(3)])}	57 passive terms

relevance feedback. The queries were as follows:

q(1) = "graph grammars programming"

q(2) = "graphical programming concurrent"

q(3) = "software tools parallel"

Table 5.1 shows the evolving context (the Active terms) and indicates the size of the passive term set after each learning session. The particular representation we have chosen for the context is a set of c-structures consisting of a stemmed term, its total term weight, the total number of documents covered, and a list of sources of the term ('m' indicates that the term was used after query expansion, and 'q' indicates the term is from the user's query).

Quite clearly the context's terms can be used to re-weight documents, and for a full evaluation of our learning algorithm we are currently aiming its use at solving the weight-block problem. Prior to this full integration with the IRS, we have gathered output from the learning implementation, such as in Table 5.1. The context formed after each iteration were shown to those more expert in the corresponding fields, and their comments on the context learned have been encouraging.

Results also seem to correspond with the improvement made to the baseline formula (equation 5.2) by equation 5.3. Using the example above, after each feedback collection, the ranks of the terms used for query expansion incorporating equation 5.2 are compared with their ranks using the improved

formula. All the terms that the learning system had acquired in the context were also ones that moved *up* in the ranked list of weighted terms between the baseline and improved formula. Hence the indications so far are that the learner is evolving contexts consistent with equation 5.3, although from a different approach. The learner also has the added advantage of using historical relevance feedback information.

5.5 Conclusions and future work

In this chapter we have specified a learning component which can use relevance feedback in a probabilistic IRS to adapt to a user's information needs. Initial testing, such as the example in section 5.4.2, suggests that the evolving context can act as a useful background to users carrying out closely related searches. The learning algorithm will be fully evaluated by applying it to solving the weight-block problem using the Okapi system. We also suspect it will be useful in more radical document re-listing, and further to decide on the set of documents to be displayed.

Possible improvements include the development of methods for removing redundant terms from the context, and the use of counter- examples to evolve the context. Finally, *user groups* with links to contexts shared by its members could be formed, suggesting terms to a user from the knowledge gained from other users' queries.