

ABSTRACT

PALATHINGAL, PAUL JOSE. Integration of Information Retrieval Techniques into a Referral System for Knowledge Management. (Under the direction of Dr. Munindar Singh.) Social networks potentially form a huge repository of private knowledge that is unavailable on any search engine. The Multi Agent Referral system (MARS) prototype is a tool for building and exploiting social networks. Referral chains represent social networks and are used to request for experts in a particular field. The MARS system uses software agents to automate the search of information and expertise.

However information about a user's expertise is not readily available. Also currently most systems do not exploit the fact that the user's document repository is a good pointer to other user's interests. By contrast this paper shows how to use Information Retrieval techniques to be able to answer queries and also make referrals on behalf of the user. TFIDF (Term Frequency Inverse Document Frequency) indexing is implemented on user documents and messages to determine user interests. A simulation has been carried out and results of the simulation are presented.

This thesis studies the effects of using term and document frequency indexing for information retrieval on user documents and messages in the MARS prototype.

**Integration of Information Retrieval Techniques
into a Referral System for Knowledge Management**

By

PAUL JOSE PALATHINGAL

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
COMPUTER SCIENCE

AT

NORTH CAROLINA STATE UNIVERSITY
RALEIGH, NORTH CAROLINA

5-20-2002

APPROVED BY:

Advisor:

Munindar Singh

Committee Members:

Peng Ning

Mona Singh

NORTH CAROLINA STATE UNIVERSITY
DEPARTMENT OF
COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Integration of Information Retrieval Techniques into a Referral System for Knowledge Management**" by **Paul J Palathingal** in partial fulfillment of the requirements for the degree of **Master of Science**.

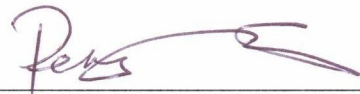
Dated: 5-20-2002

Advisor:

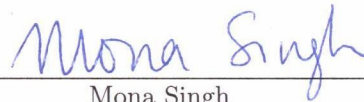


Munindar Singh

Committee Members:



Peng Ning



Mona Singh

DEDICATION

To my late grand father P O Poullose.

To my parents who have made my higher education dreams come true .

BIOGRAPHY

Paul Jose Palathingal was born in Trichur, Kerala, India. He did his undergraduate program at Government Engineering College, Trichur, India and received a BTech in Computer Science and Engineering from University of Calicut in 1997. He joined North Carolina State University in the fall of 1999 to pursue an MS in Computer Science. He worked as a Research Assitant for Dr. Mumnindar Singh in the field of Agents and Information Retrieval for about two years. As part of his MS co-operative program, he has worked at IBM (Websphere integration) at the Research Triangle Park, Raleigh.

ACKNOWLEDGEMENT

I would like to thank Dr Munindar Singh, my supervisor, for his many suggestions and constant support during this research. I am also thankful to Bin Yu and Wentao Mo for their guidance through my years of chaos and confusion.

Dr Singh was instrumental in bringing me to the field of Multi-agent Systems. I would like to thank him for his concerted efforts towards my thesis. Bin Yu supplied me with material on some of his recent joint work with Dr Singh, which gave me a better perspective on my own results.

I would like to thank Dr Peng Ning and Dr Mona Singh for serving on my thesis committee.

I am grateful to my parents for their patience and *love*. Without them this work would never have come into existence (literally).

Finally, I wish to thank the following: Harish (for his friendship); Dhruva (for changing my life from worse to bad); all my friends who have helped me with the Thesis *and* my sisters (because they asked me to).

This work was partially supported by the National Science Foundation through grant ITR-0081742.

Paul J Palathingal

May 20, 2002

© *Copyright by Paul J Palathingal, 2002*

Table of Contents

List of Figures	viii
1 Introduction	1
1.1 Knowledge Management	1
1.2 Expertise Location	2
1.3 Traditional Information Retrieval	4
2 Multi Agent Referral System	6
2.1 MARS Prototype	6
2.2 MARS Components	8
2.3 Motivation For Information Retrieval	10
3 Indexing	14
3.1 Introduction	14
3.2 Scanning The Text	16
3.3 Vector Space Models	17
4 Information Retrieval In MARS	21
4.1 Information Retrieval In MARS	21
4.2 The Indexer Architecture	22
4.3 Referral Scenarios In MARS	25
5 Implementation	29
5.1 The Class Diagram	29
5.2 Analysis	32

6 Discussion	35
A Package Structure	37
Bibliography	45

List of Figures

2.1	Referral Process	11
2.2	Example Ontology	12
2.3	Multi Agent Learning	13
3.1	Overview Of Search And Index Mechanisms	15
4.1	Indexing Architecture	26
4.2	Scanning documents	27
4.3	Indexing documents	27
4.4	Similarity Check	28
5.1	Indexer Module Collaboration Diagram	30
5.2	Indexer Module Class Interaction	34
A.1	Package engine Class Diagram	37
A.2	Package engine.kb Class Diagram	38
A.3	Package engine.learner Class Diagram	39
A.4	Package engine.reasoner Class Diagram	40
A.5	Package engine.referral Class Diagram	41
A.6	Package ui Class Diagram	42
A.7	Package mail Class Diagram	43
A.8	Package indexer Class Diagram	44

Chapter 1

Introduction

Information Retrieval (IR) is a wide, often loosely defined term [Groven, 1995]. The word information can be very misleading. In fact, in many cases one can adequately describe the kind of retrieval by simply substituting “document” for “information”. Information Retrieval does not inform the user on the subject of his inquiry, but only merely informs on the existence and whereabouts of documents relating to his request. Simply put, an Information Retrieval system leads the user to those documents that will best enable him or her to satisfy his or her need for information. We begin with a look at knowledge management, referral systems and IR techniques used.

1.1 Knowledge Management

Knowledge management is a process that helps organizations find, select, organize, disseminate, and transfer important information and expertise necessary for problems such as problem solving, dynamic learning, strategic planning and decision making [Gupta et al., 2000]. Polanyi first distinguished between tacit and implicit knowledge [Polanyi, 1958].

Tacit knowledge is usually in the domain of subjective, cognitive, and experimental learning, whereas explicit knowledge deals with more objective, rational, and technical knowledge. The above two types of knowledge can be seen as relating to the following classes of approach for knowledge management:

- Expertise location.
- Traditional information retrieval.

The approach developed here considers the first, but applies some techniques from the second.

1.2 Expertise Location

Significant research has been done on referral systems and social networks. Interpersonal communication acts as an important channel for gathering information [Yu et al., 1999]. For example, suppose you need to see a dentist, and want to know if there is a good dentist near where you live. You might find information on dentists in your area when you look up the Web. However, the search will not tell you how good he is or how experienced he is. Experts in a particular field are not known directly. Through friends or neighbors we usually reach a potential expert. Six Degrees Of Separation [Singh, 1999] is a phenomenon that states that large social networks can have remarkably short paths between participants. For the case of multiagent systems, it gives a strong reason to believe that intelligent software agents may be able to interpret links to other agents and follow only the relevant ones, so as to find the desired experts quickly.

The idea of using referral systems to model social networks has appeared only recently.

MINDS is one of the earliest agent-based referral system [Salton and Buckley, 1988]. ReferralWeb is another referral system that is based on co-occurrence of names on web pages [Kautz et al., 1997]. However, none of these use interpersonal communications or the social structure to determine plausible referrals. The prototype system MARS, however, is developed to be used in a practical social network. The architecture is fully distributed and each agent learns models of each other in terms of expertise (ability to answer queries effectively) and sociability (ability to give referrals effectively).

The MARS system is an active model aimed at achieving an effect similar to the phenomenon of the Six Degrees Of Separation. The model is distributed in its functioning. MARS distinguishes between user interests and user expertise. A user will query based on his interest and will respond to a query based on his expertise. Determining which user or neighbor to query is done by incorporating a Vector Space Model to social networks. Queries, user interest, expertise are all represented by vectors. Applying similarity based on Term Frequency Inverse Document Frequency (TFIDF) algorithms, we determine the user whose expertise is the most similar to the query.

To adapt to finding information that is more private and informal, the MARS system has incorporated the Information Retrieval system outlined in this Thesis. When a user formulates a query, the user agent has to determine which contacts to send the query to. Using the Vector Space model an agent is able to estimate the importance of each term in a query and use it to discriminate among the users who may be sent that query based on the expertise they have exhibited [Yan and Garcia-Molina, 1993].

1.3 Traditional Information Retrieval

Mankind today has to handle both dynamic and static knowledge structures. For example in the field of Web based training particular domain knowledge can be built by the course material [Dietinger et al., 1999]. The dynamic part may consist of relevant web sites or web areas, new forums etc. The static library will include electronic books, electronic journals, exercises, student papers etc. Static as well as dynamic knowledge sources are gathered, processed and stored by a knowledge processor. An indexer system could implement such a knowledge processor. Experience has shown that only relevant terms depending on the proper domain subject or environment have to be taken into account for describing and clustering documents. An indexing system does just that by clustering documents based on term and inverse document frequencies. This is a traditional information retrieval approach and helps extract implicit knowledge.

Conventional Information Retrieval has been based on existing Boolean text search systems. In such a system, search requests are normally represented by Boolean statements, consisting of search terms interrelated by the Boolean operators and, or, and not. The retrieval system selects those stored items that are identified by the exact combination of the search items specified. Our approach used indexing mechanisms to describe documents and requests. The elements of the index language are index terms, which may be derived from the text of the document to be described, or may be arrived at independently. The index terms are weighted as a function of the rank order of their frequency of occurrence. There are two types of weighting, one in which the weight is given to a term based on the number of documents in which it occurs (Document Frequency) and the other based on the frequency of occurrence of a term in a particular document (Term Frequency).

The difference between the Document Frequency weighting and the Term Frequency

weighting may be summarized by saying that document frequency weighting places emphasis on content description whereas term frequency attempts to emphasize the ability of terms to discriminate one document from another. The Term Frequency Inverse Document Frequency indexing scheme uses the two features to extract information from a collection of documents accurately. The TFIDF scheme is integrated with the MARS system to search for potential experts and accurate location of expertise by an agent.

This thesis discusses how to combine these two aspects of knowledge management by integrating IR techniques into a referral system. The IR techniques apply to documents to bootstrap the referral system.

Chapter 2

Multi Agent Referral System

In this chapter we look briefly at the Multi Agent Referral System (MARS), and the motivation for this thesis.

2.1 MARS Prototype

MARS is an agent based system that combines techniques from information retrieval, multi agent learning and adaptive user modelling to refine the social network to the users needs [Yu and Singh, 2000]. Unlike many previous approaches, MARS is fully distributed and preserves the privacy and autonomy of its users. To better understand the referral function in the MARS system refer to Figure 2.1

There are basically two types of agents - an agent who sends out a query and an agent who answers queries or gives referrals. All the agents in the MARS system have the following:

- User Profile: This profiles the users expertise areas.
- Neighbor model: A model or shortcut to neighbors that are in the user's realm. The neighbor model contains the neighbors expertise and sociability (which is a measure of how well the neighbor is able to refer other potential agents).

A querying agent sends out a query to a set of neighbors based on their expertise. The originating agent can receive two types of replies. An answer, in which case the agent evaluates the answer and updates the neighbor model. A referral in which case the agent sends out the query to the referred agent. If an answer is a good one, the agent updates the expertise of the answering agent to reflect that and vice versa. Also if the referral is a good referral the sociability of the referring agent goes up and the expertise value of the referred agent gets a lift.

An answering agent can take either of three actions:

- Answer: Answer with respect to the query based on the expertise of the user or agent.
- Refer: Based on the neighbor model, the agent can refer another potential expert to the querying agent.
- Ignore: The agent can totally ignore the query.

2.2 MARS Components

- **Ontology** - Knowledge and expertise values learnt from users and agents are represented using an ontology. An Ontology is a specification of a conceptualization [Gruber, 1993]. Practically, an ontological commitment is an agreement to use a vocabulary in a way that is consistent with respect to the theory specified by an ontology. In the MARS system, an ontology is used to represent a query domain and the user's expertise. There are 30 fields in the domain of AI programming used in the MARS ontology.
- **Neighbor Model** - The neighbor model is a list of agents whom the user's agent knows. The reason the MARS system has a neighbor model is two fold.

When a user is sending a query, the user's agent must decide to whom from the neighbors list the query will be sent to.

For an incoming query, if the user does not have the expertise to answer it, the agent uses the neighbor model to decide which agents in the neighbor list the query will be referred to. Refer to Figure 2.2 for an example ontology.

To do the above each neighbor model must contain

- **Expertise vector** - The neighbors expertise on different domain fields (each value is between 0 and 1).
- **Sociability** - The value which is a scalar is a measure of the neighbors sociability level, or in other words how well the neighbor knows other expert neighbors. Also when a user asks a question and gets an answer from someone who is neither in the close friend list nor in the neighbor model, and if the user evaluates the answer as a good one, the agent will be added to the neighbor model.

- Multi Agent Learning - Each agent stores information about the user and his close friends (expertise, sociability) during the bootstrap period of the system. After that, the agent will maintain and modify the information through the interactions between the user and his agent, and the agent and other agents. Please refer to Figure 2.3.

When a good answer is obtained, the expertise of the agent who provides the answer will increase, and so will the sociability of the agents who made referrals if any. On the contrary if a bad answer is obtained, the expertise of the answering agent will decrease and so will the sociability of the referring agents, if any.

- Referral Graph - Being able to make Referrals is the core to the whole MARS system. For a specific query, it may be referred by a number of agents before it is answered. To keep track of the referring agents and to maintain a referral chain corresponding to the query, the MARS system needs a referral graph.

2.3 Motivation For Information Retrieval

The MARS system rides on its referral capability. This would imply locating expertise necessary to solve difficult problems. In organizations, some people assist others in locating expertise by making referrals. People who make referrals fill key organizational roles. The motivation behind using IR in the MARS system is to decrease workload and support users by automatically making expert referrals. To be able to make referrals in an IR setting is more of a recommendation system. However combining the referral power of MARS with the recommendations of the IR system to make the user more autonomous, was one of the motivations to use

Another reason that is that in organizations, an IR module in the MARS system leads to a system, which is a possible technology that can augment and assist the natural expertise locating behavior. A referral system that suggests people who have some expertise with a problem holds the promise to provide, in a very small way, a service similar to that provided by key people.

In the current MARS prototype, the referral agent passes on queries to the user. And based on decisions from the expertise values of neighbors, the user can suggest referrals. There is scope here for the agent to provide the referral directly, without intervention from the user. To test the system, two different approaches can be pursued. One where the referral is made only by the agent, based on what he knows about the users neighbors. And the other where both the agent and user make referrals based on their knowledge of the users neighbors.

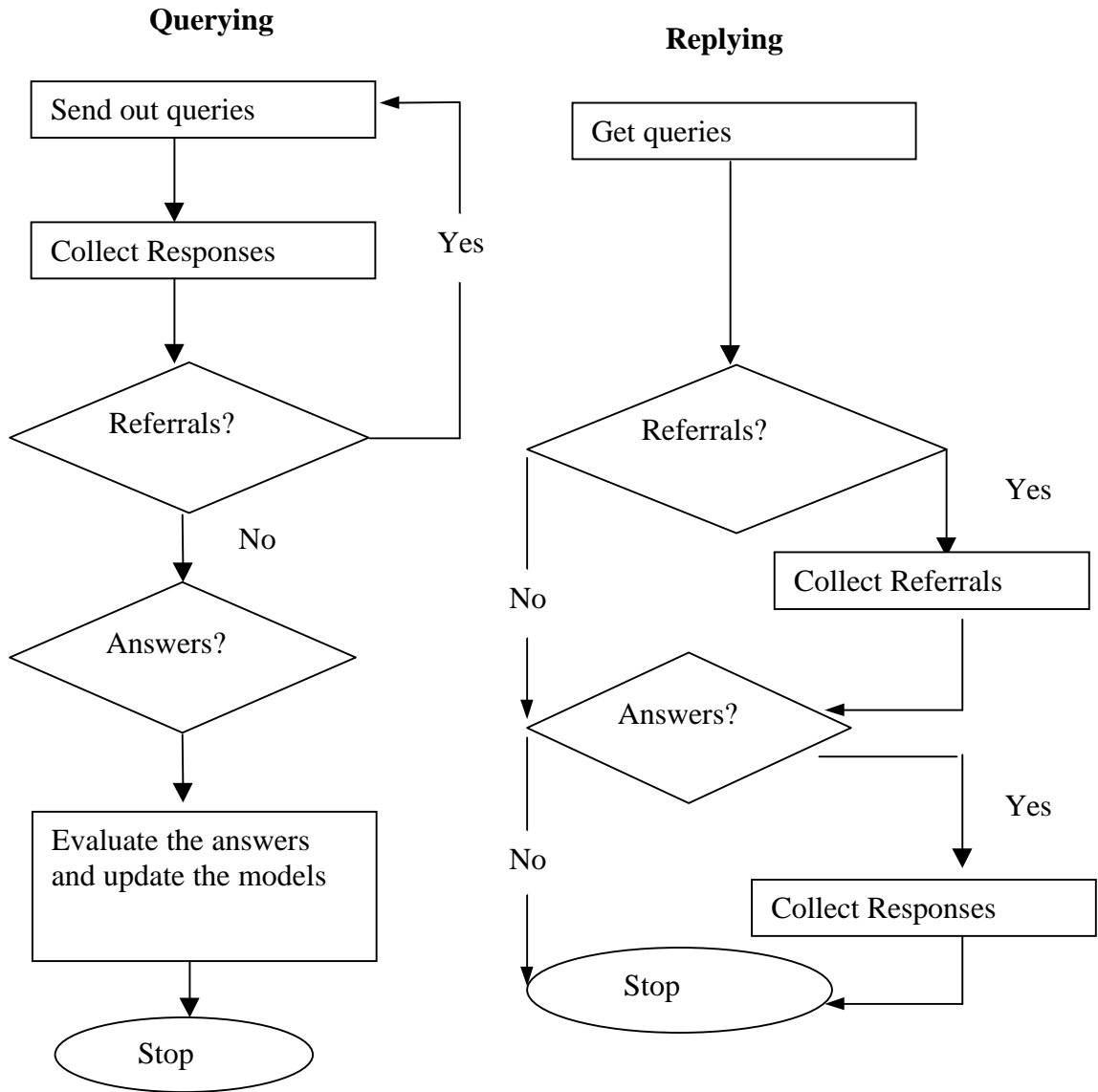


Figure 2.1: Referral Process

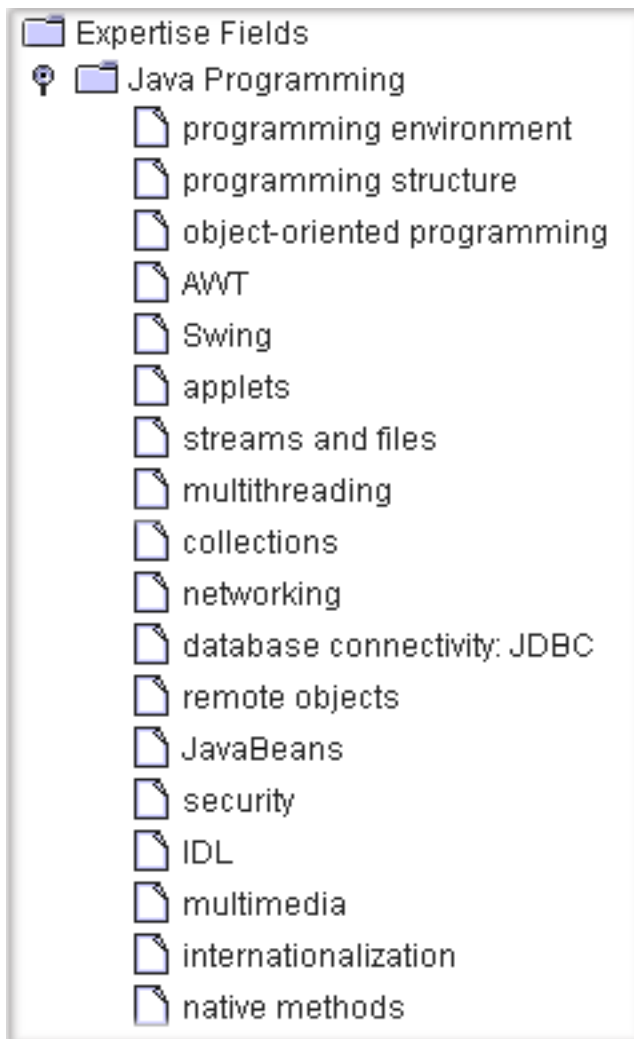


Figure 2.2: Example Ontology

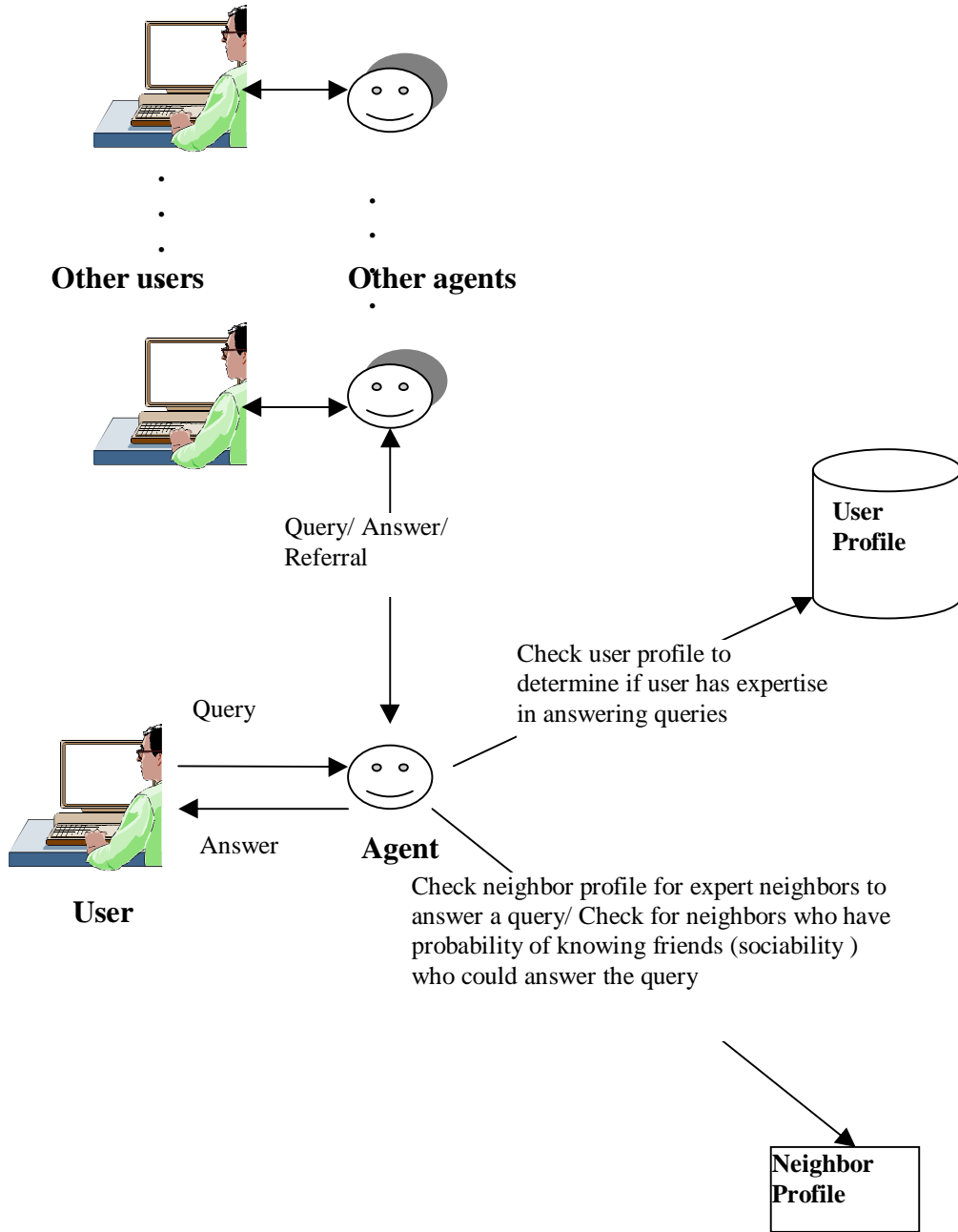


Figure 2.3: Multi Agent Learning

Chapter 3

Indexing

3.1 Introduction

Intuitively, an indexing scheme is simply a collection of blocks, each block containing some large, fixed number of objects (typically hundreds of objects per block) [Brown et al., 1994]. The union of the blocks exhausts the instance. Each query is answered by retrieving a set of blocks whose union is a superset of the query [Hellerstein et al., 1997]. Simply put it means that given a set of documents, the process of finding an exhaustive set of keywords that identify the text in the documents and storing these terms in some structure is called indexing. Indexing addresses the issue of how information from a collection of documents should be organized so that queries can be resolved efficiently and relevant portions of the data extracted quickly. An index must be capable of identifying all documents that contain combinations of specified terms, and the process of identifying the documents based on the terms is called a search or query of the index.

Given a cluster of documents it can be time taking to come up with an index file that appropriately indexes the cluster. The indexing mechanism is broadly divided into the following:

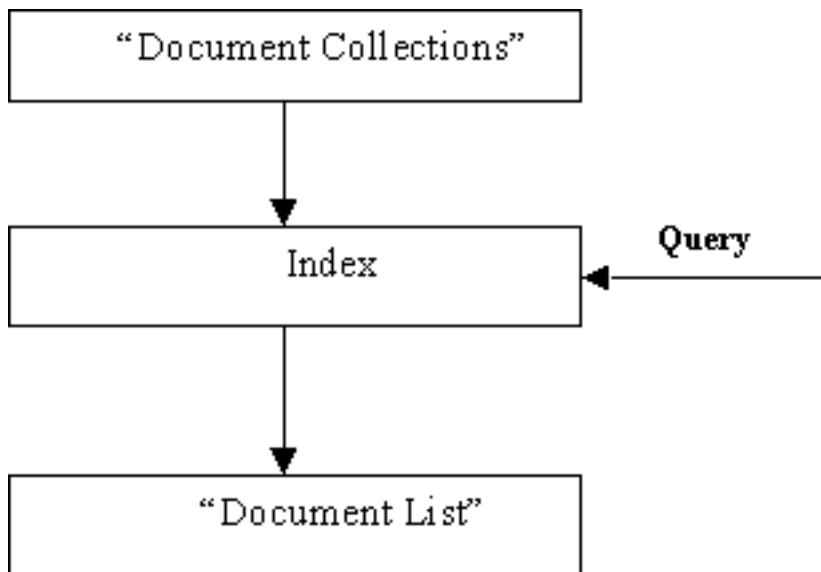


Figure 3.1: Overview Of Search And Index Mechanisms

- Full text Scanning.
- Vector Space Model.

3.2 Scanning The Text

Scanning the text is a very time consuming process. Hence a considerable amount of work has gone into it. To make a set of documents available for indexing, you must first make a collection. A collection is a set of directories and files that allow you to quickly find and display source documents that match various search criteria. To make sure that you have the right collection, the process of gathering the collection of documents follows a distinct process. The documents are collected based on the need of the agent to avoid indexing a huge collection. This would speed up the search and help in returning documents that are more accurate and closer to the query. An index table is created, and the name of the term, name of the document in which it occurs and the number of times the term occurs in each document is stored.

The text is scanned for key terms [Alexander and McCracken, 1994]. While scanning, there are four types of lexical entities that are processed as key terms:

- Proper nouns.
- Proper noun categories - Before indexing occurs, proper nouns have already been identified and placed in categories such as city, company, person, etc.
- Complex nominal's - Before indexing occurs, phrases consisting of a combination of nouns and adjectives (i.e. trade ban, chemical dependency seminar) are tagged as complex nominal's.
- Single words - nouns, verbs, and adjectives.

3.3 Vector Space Models

The vector-space models for information retrieval are just one class of retrieval techniques that have been studied in recent years. We could term it as a partial match retrieval technique since they typically rely on an underlying formal mathematical model for retrieval, model the documents as sets of terms that can be individually weighted and manipulated, perform queries by comparing the representation of the query to the representation of each document in the space, and can retrieve documents that don't necessarily contain one of the search terms [Berry et al., 1995]. Although the vector-space techniques share common characteristics with other techniques in the information retrieval hierarchy, they have a core set of functionality's that justify their own class.

Vector-space models rely on the premise that the meaning of a document can be derived from the document's constituent terms [Letsche and Berry, 1997]. They represent documents as vectors of terms $d = (t_1, t_2, \dots, t_n)$ where $t_i (1 \leq i \leq n)$ is a non-negative value denoting the single or multiple occurrences of term i in document d . Thus, each unique term in the document collection corresponds to a dimension in the space. Similarly, a query is represented as a vector $q = (f^1, f^2, \dots, f^n)$ where term $f^i (1 \leq i \leq n)$ is a non-negative value denoting the number of occurrences of f^i (or, merely a 1 to signify the occurrence of term f^i) in the query. Both the document vectors and the query vector provide the locations of the objects in the term-document space. By computing the distance between the query and other objects in the space, objects with similar semantic content to the query presumably will be retrieved.

Vector-space models that don't attempt to collapse the dimensions of the space treat each term independently, essentially mimicking an inverted index. However, vector-space models are more flexible than inverted indices since each term can be individually weighted,

allowing that term to become more or less important within a document or the entire document collection as a whole. Also, by applying different similarity measures to compare queries to terms and documents, properties of the document collection can be emphasized or de-emphasized. For example, the dot product (or, inner product) similarity measure finds the Euclidean distance between the query and a term or document in the space. The cosine similarity measure, on the other hand, by computing the angle between the query and a term or document rather than the distance, de-emphasizes the lengths of the vectors.

The TFIDF approach to document classification works as follows:

- Let V be the vocabulary used. Let d be a document. The document is processed using stemming and stopping procedures to obtain a bag of words for document d .
- Let w_i be the i^{th} word in the vocabulary V .
- Term frequency of w_i , $TF(w_i, d)$ is the number of times w_i occurs in d .
- The document frequency of w_i , $DF(w_i)$, is the number of documents in which w_i occurs at least once.
- Inverse document frequency of w_i , $IDF(w_i)$ is defined as $IDF(w_i) = \log \left(\frac{|D|}{DF(w_i)} \right)$, where $|D|$ is the total number of documents.
- Then, the term frequency - inverse document frequency of w_i , $TFIDF(w_i, d)$ is given by $TF(w_i, d) * IDF(w_i)$.
- The vector representation \vec{d} of a document d is given by $\vec{d} = [TFIDF(w_1, d), TFIDF(w_2, d), \dots, TFIDF(w_n, d)]$.

Vectors are normalized by their lengths in order to be able to cope with documents of differing lengths. This is accomplished by computing the length of the vector representing the document and dividing the weights of the terms by this value. Hence the vector \vec{d} is given by $\vec{d} = [V_1, V_2, \dots, V_n]$, where $V_i =$

$$\frac{TFIDF(w_i, d)}{\sqrt{\sum_i TFIDF(w_i, d) * TFIDF(w_i, d)}}$$

The queries in our model are expressed in a way similar to that documents are expressed, i.e., a query Q with k terms is represented by the vector $\vec{V} = [TFQ(w_1), TFQ(w_2), \dots, TFQ(w_n)]$. The weights assigned to the terms will describe the importance of each term. Unlike the way the weight of a document term is calculated, the weight of a query term will be assigned solely based on the frequency of that term in the query.

The ranking of a query with respect to a document d is done by using the cosine similarity given below

$$\frac{\sum_i TFIDF(w_i, d) * TFQ(w_i)}{\sqrt{\sum_i TFQ(w_i) * TFQ(w_i)} \sqrt{\sum_i TFIDF(w_i, d) * TFIDF(w_i, d)}}$$

When a query is posed to the IR system, the system parses the query first, extracting its terms, and then evaluates the query by calculating its relevance to the documents in its document collection. If there is no index on the documents, then all the document vectors must be processed in order to find the ones that relate highest to the query. This is a very expensive process as there are typically millions of documents and each document typically contains at least 100 to 1000 terms (which defines the number of floating point operations to be done while calculating the relevance of a query and a document by using the relevance

measure as described above).

In order to evaluate user-queries efficiently, all IR systems make use of inverted indexes. An inverted index contains, for each term that occurs in the documents of the document collection, a list of $\langle DID, f_{t,DID} \rangle$ pairs where DID is the unique identifier for the document that contains the term, and $f_{t,DID}$ is the frequency of the term in that document. The usage of an inverted index allows to reduce the number of documents to be processed by processing only those documents that contain at least one term from the query.

Chapter 4

Information Retrieval In MARS

In Chapter 2 we saw the MARS prototype briefly introduced. Now we shall go into the details of the system and how an effective information retrieval system based on the TFIDF indexing scheme can be incorporated.

4.1 Information Retrieval In MARS

This is a brief introduction to how the indexing and information retrieval mechanism discussed in Chapter 3 can be applied to the MARS system. Vector space models and corresponding information retrieval methods have been used in many news and information filtering systems recently. The MARS system adapts the vector space model to locate people rather than documents. The way that this can be achieved is by modelling users - User Profile and modelling other users - Neighbor model. Vectors are used to represent user expertise values and also expertise values of each neighbor. This vector is called the expertise vector. From Chapter 3, we saw that when we are able to model an entity as a vector, it is possible to do a similarity measure determination to a particular query. This feature can be extrapolated and used in the MARS system.

4.2 The Indexer Architecture

A look at the indexer architecture in (Figure 4.1) lays out the following

Components:

- **Scanner** - The scanner as seen in Chapter 3, scans all documents and looks for terms that could be indices to the documents. The criteria for scanning such terms or texts are as laid out in Chapter 3. The result of the scanner is sent to the indexer.
- **Indexer** - The indexing mechanism used is the TFIDF indexing scheme. The scanned terms are used as indices to the document list. The indexer determines the document frequency and the term frequency as defined in Chapter 3. Based on this a TFIDF vector and index table is created.
- **Similarity Check** - This module compares 2 vectors based on the dot product rule and returns the documents that have a dot product value greater than or equal to a threshold. The results are sent to the result processor.

Processes:

- **Scanning Documents** - Most of the indexing process is done independently of the rest of the system. The first step in the indexing process, scanning of documents is done every 5 minutes. The scanned words also called the index terms are passed back to the Indexer.
- **Indexing Documents** - Once the documents are scanned for index terms, the indexer indexes into the documents. This is done every 5 minutes and is again independent of the rest of the system. This process returns the TFIDF vector for the document list.

- Build - This process builds the index table that will be used to determine relevance of documents to queries. The process returns an index table that is stored.
- Save - When a query is matched to relevant documents in the document list, this match is saved to create a log of recent queries and possible matches.

Data Collections:

- Query List - Queries are queued up at the input to the indexer. This list is stored as a queue and served on a first come first serve basis.
- Index Table - The indexing process returns the TFIDF vector which is stored as an index table for future similarity check to relevant documents.
- Document List - This data collection includes user documents that profile his or her expertise in a certain field. It also contains e-mail logs of queries and answers.
- Query Search Results Log - This is a form of a cache that can be quickly looked up in the future if a similar query comes to the agent.

The indexer control flow is divided into the following

- Scanning documents: Please refer to Figure 4.2.
- Indexing documents: Please refer to Figure 4.3.
- Query matching: Please refer to Figure 4.4.

Each user has a set of documents that model his or her expertise. For example an author in the field of artificial intelligence is bootstrapped with a set of documents in the artificial intelligence domain. These documents may be files, html pages, xml documents. Each

time a new document is added to the users list of documents, it becomes an integral part of the indexed documents.

Each user is also assigned a referral agent. The agent as seen in Figure 4.1 has a user profile and a neighbormodel. It also indexes the user documents. Every time a new document is added it gets indexed and the index table is updated.

4.3 Referral Scenarios In MARS

There are two possible scenarios of implementing the IR module with the existent MARS system. They differ in the fact that one maintains greater referral capability than the other.

- Scenario 1: When an agent receives a query, it is indexed and compared to the indexed user documents. If there are matches the documents are sent as replies. If there is no match the query is passed on to the user. Based on his or her neighbor model, the querying agent is referred to potential experts. The problem with this scenario is that in many situations the referral power of the system is lost. However it provides greater autonomous nature to the user.
- Scenario 2: When an agent receives a query, it is indexed. It is passed on to the user for potential referrals, and also expert replies. At the same time, the query is compared to the indexed documents. The answer here consists of both user specific answers like answers and referral, and matched indexed documents. All replies are sent back to the requesting agent. The advantage with this scenario is that the referral power of the system is maintained.

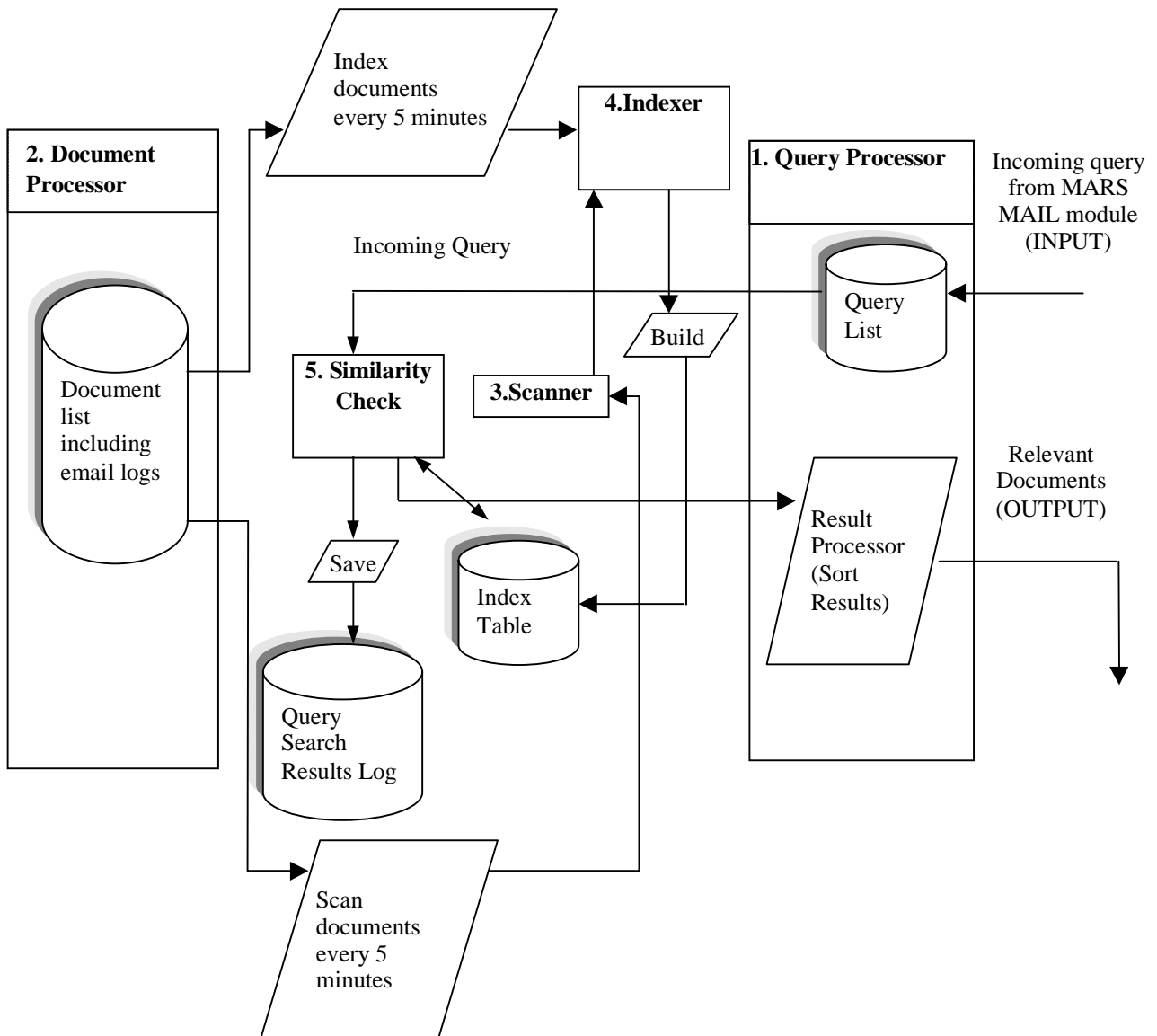


Figure 4.1: Indexing Architecture

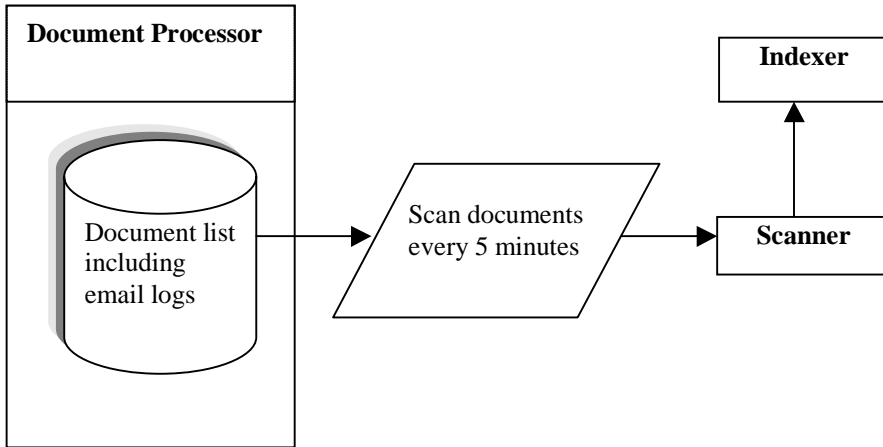


Figure 4.2: Scanning documents

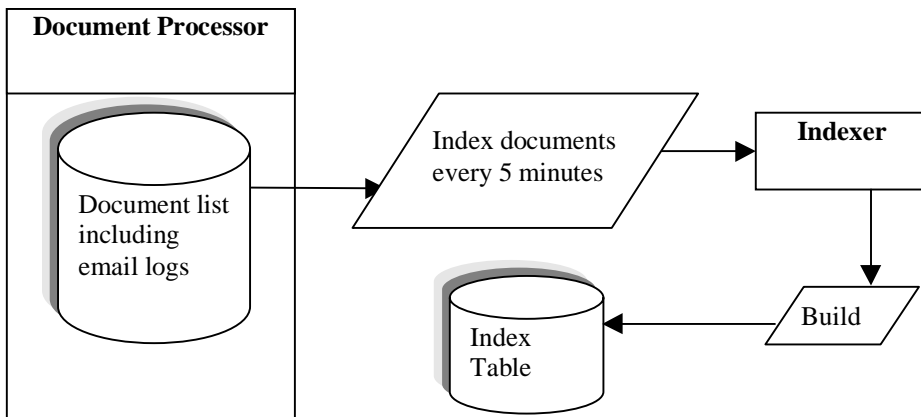


Figure 4.3: Indexing documents

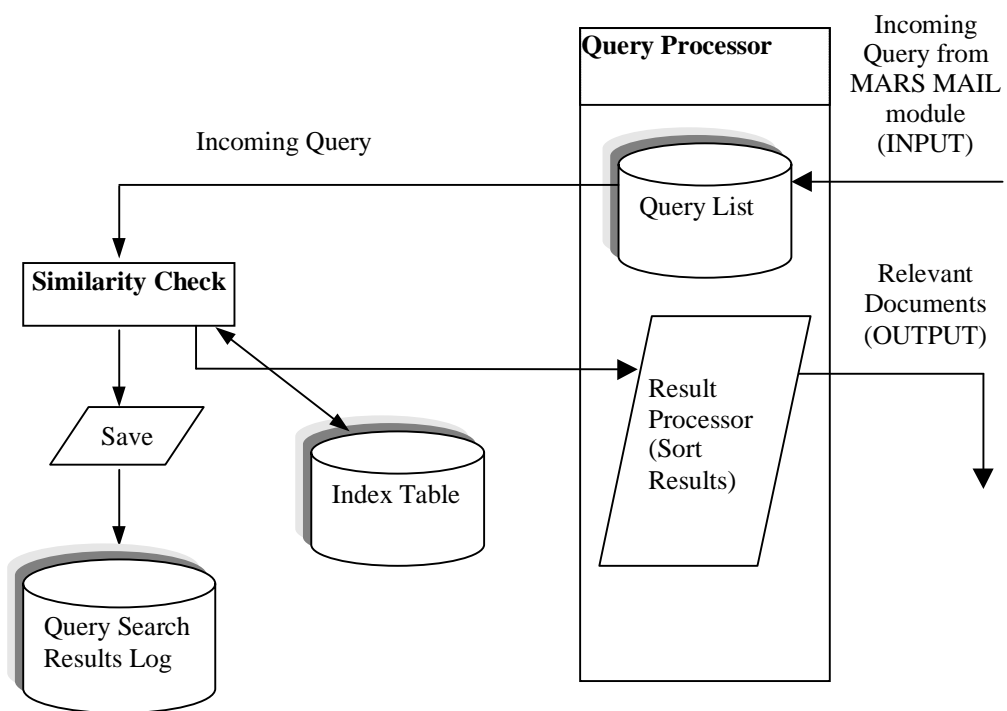


Figure 4.4: Similarity Check

Chapter 5

Implementation

5.1 The Class Diagram

The implementation of the indexer involves using some programming tools and utilities. Sun's Java Mail API, POP3 is the mail interface used and the indexer is written in Java using Java 1.2. Here is a listing of the MARS packages

- engine - This package implements the learning, reasoning and referral processes in each agent. Refer to Appendix A.1 for the class diagram.
- engine.kb - Refer to Appendix A.2 for the class diagram.
- engine.learner - Refer to Appendix A.3 for the class diagram.
- engine.reasoner - Refer to Appendix A.4 for the class diagram.
- engine.referral - Refer to Appendix A.5 for the class diagram.
- ui - MARS interfaces that enable the interactions between the user and MARS, i.e., transforming the queries and responses between the Reasoner, Learner and the user. Refer to Appendix A.6 for the class diagram.

- mail - This package deals with sending and receiving message between agents. Refer to Appendix A.7 for the class diagram.
- indexer - This package implements an indexer that indexes messages and documents. Refer to Appendix A.8 for the class diagram.

A collaboration diagram of the interaction between the indexer classes is shown in Figure 5.1

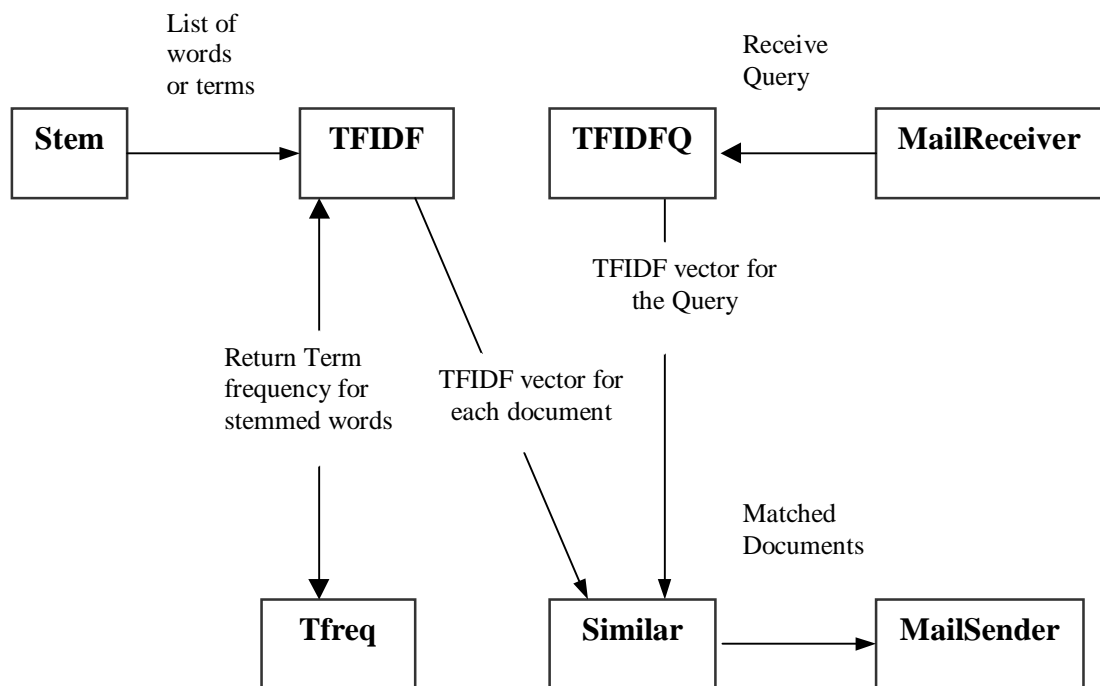


Figure 5.1: Indexer Module Collaboration Diagram

A class diagram of the interaction between the indexer classes is shown in Figure 5.2

The control flow is as follows:

1. The MailReceiver class accepts incoming queries and forwards them to the agent.
2. The TFIDFQ class determines the Term Frequency and Inverse Document frequencies for the incoming query.

- 3. The Stem class is a stemmer that deletes stop words from the documents and generates the index terms. This can be done in parallel with steps 1 and 2.
- 4. TFIDF calls the Tfreq class to determine the frequency of occurrence of each term in each document. The inverse document frequency, that is the number of documents that contain the each term is also returned.
- 5. To determine or search for documents that match the incoming query, TFIDF and TFIDFQ pass their TFIDF vectors to the Similar class. The Similar class does a dot product on the two vectors and returns to the MailSender class a set of matching documents.

5.2 Analysis

Test for referrals

- 1. Select five authors from the Artificial Intelligence domain and assign one author to one MARS agent.
- 2. Index abstracts to their proceedings on the AAAI (American Association for Artificial Intelligence) forum.
- 3. If an abstract has co-authors, assign them as neighbors, and bootstrap an expertise level for each of them.
- 4. Tests:
 - 4.1 Test the case where: Kautz is looking for expertise on planning. Kautz and Selman are co-authors. Selman has Levesque as his neighbor, who is an expert in planning. Hence test the referral mechanism: Kautz \rightarrow Selman \rightarrow Levesque.
 - 4.2 Test the case where: Kautz is looking for expertise on combinatorial auctions to supplement his research. Kautz and Selman are co-authors. However Selman has no expertise in auctions, and has no neighbors in its neighbor model to refer to. Possibly set up a registration server.
 - Result: It was seen that by doing the above experiment, a referral chain was established. Kautz was able to locate Levesque who is an expert in planning using the referral chain.
 - : The above experiment was then done for 5 authors and it was seen that the social network converged quickly through the referral chains that were established between each of the authors.

Test how the Kautz → Registration server → Sandholm (who is an expert in Combinatorial auctions work)

Test for document types

Test how the system works with different types of documents (HTML, XML, TEXT)

It was seen that the MARS system worked efficiently for HTML, XML and Text documents. HTML and XML documents were indexed based on their tags.

Contrast different scenarios

The two scenarios in 4.2, where the agent passes the query to the user and waits from the indexer to pass it to the user above were contrasted. It was seen that the system converged faster when the query was passed to the user. Or in other words when the user had less autonomy the system converged faster.

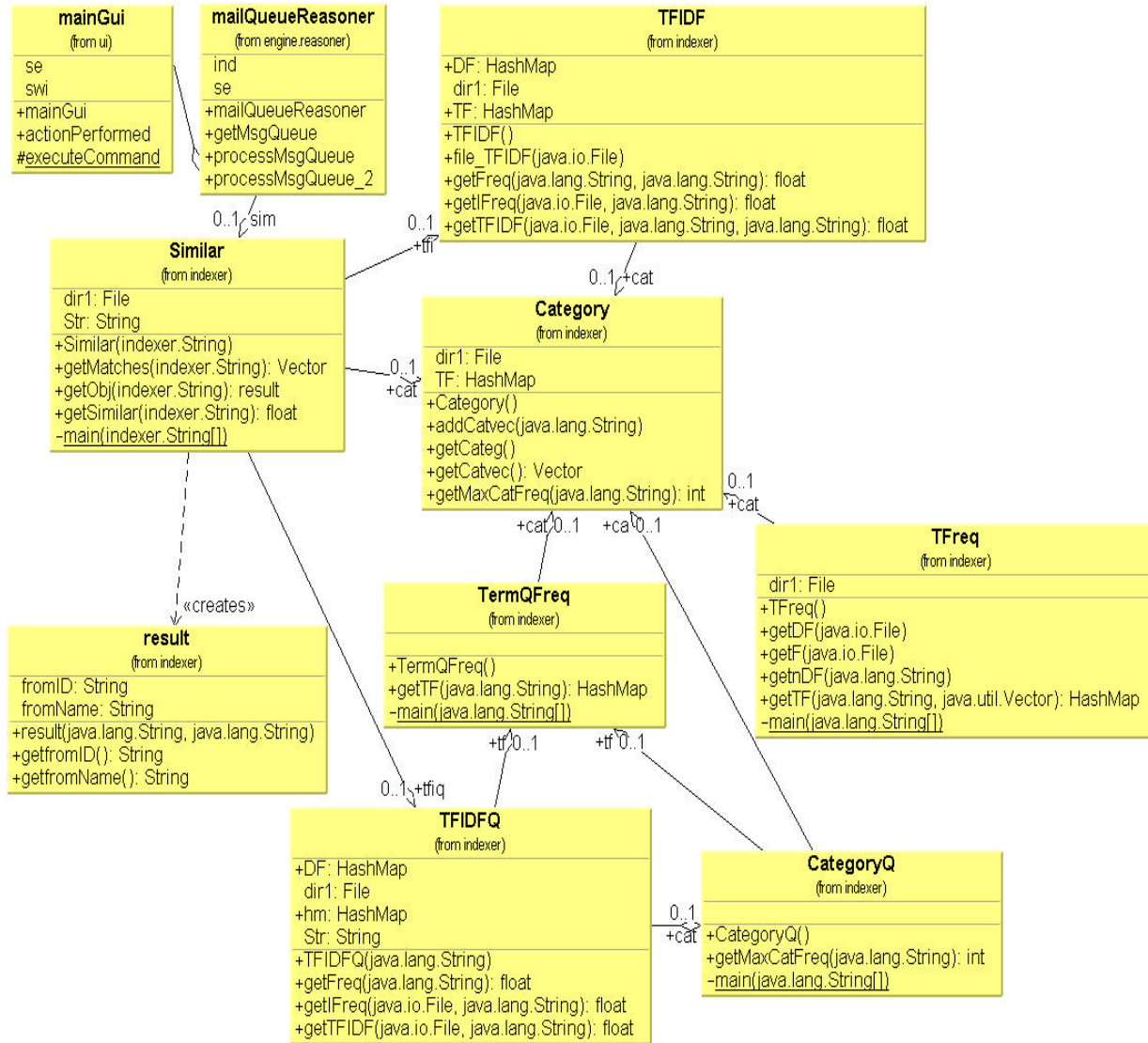


Figure 5.2: Indexer Module Class Interaction

Chapter 6

Discussion

- Semantic Indexing - The current indexing is based on term frequencies and word matching. However in the future this could be taken one step further to build a semantic indexer. Using statistical techniques to index collections for deeper search than word matching has its advantages in leading to better and more accurate search results. It would involve coming up with semantic indexes that record the correlation of noun phrases, and could be computed generically, independent of subject domain.
- Document Analyzer - Though the query was generated by the user, it might still be possible to evaluate answers independently. This comes to light more so when the answering agent returns documents to the querying agent. It can be foreseen to have an analyzer at the querying agent determine the relevance of the returned document. It might also be a way to determine the expertise level of the answering agent. It is not a sure shot guarantee to determine how relevant a query is to a given document.
- A registration server will be introduced to help the agents new to the MARS community, or in case the agent does not have any helpful neighbors for its query or no response for all agents being asked. The agent will periodically update its profile information on the server. At the same time, the agent can decide if it wants its profile

information visible to the community.

The registration subsystem will be implemented in a conventional manner. The first tier will use a Web browser or a MARS application. For the first time, the user will need to register with the server by visiting MARS web site (the user's profile is saved on the database). After the user downloads MARS, the MARS application will connect to the Application Server and retrieve or update the corresponding information.

The second tier will be implemented with a Web server running Java servlets. The Java servlet is able to access the database (via JDBC) and return an HTML page listing the data or store the information in Enterprise JavaBeans (EJB).

The third tier will be the back-end database server. The Java servlet can use information in the database provided that a JDBC driver exists. In our situation, we will use MS-Access so we can use the JDBC-ODBC driver that is bundled with the Java Development Kit versions 1.1 and higher.

- Query Expansion - Queries can be expanded to include a subset of a domain than is provided in the ontology. For example suppose a query "what is JAVA" is sent out under the Computer field in the ontology. A query expansion technique could be added to make the ontology "what is JAVA" under the Language field of the ontology. A mechanism to expand queries would greatly enhance the probability to be able to match that query with a particular users interest.

Appendix A

Package Structure

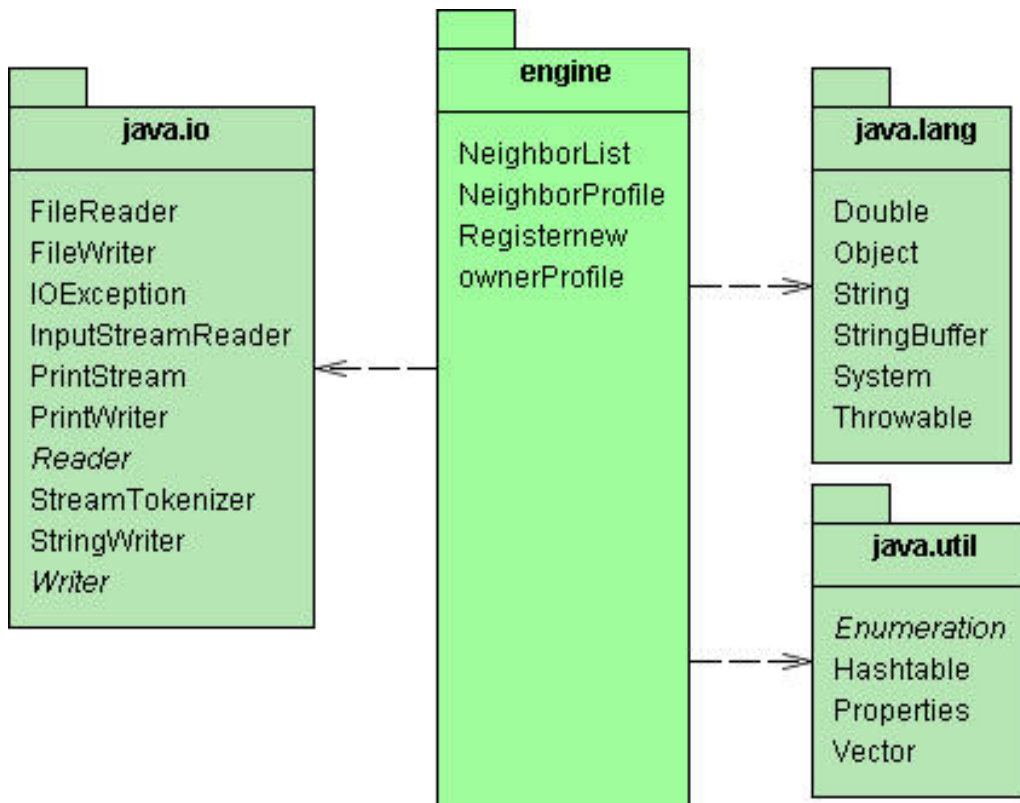


Figure A.1: Package engine Class Diagram

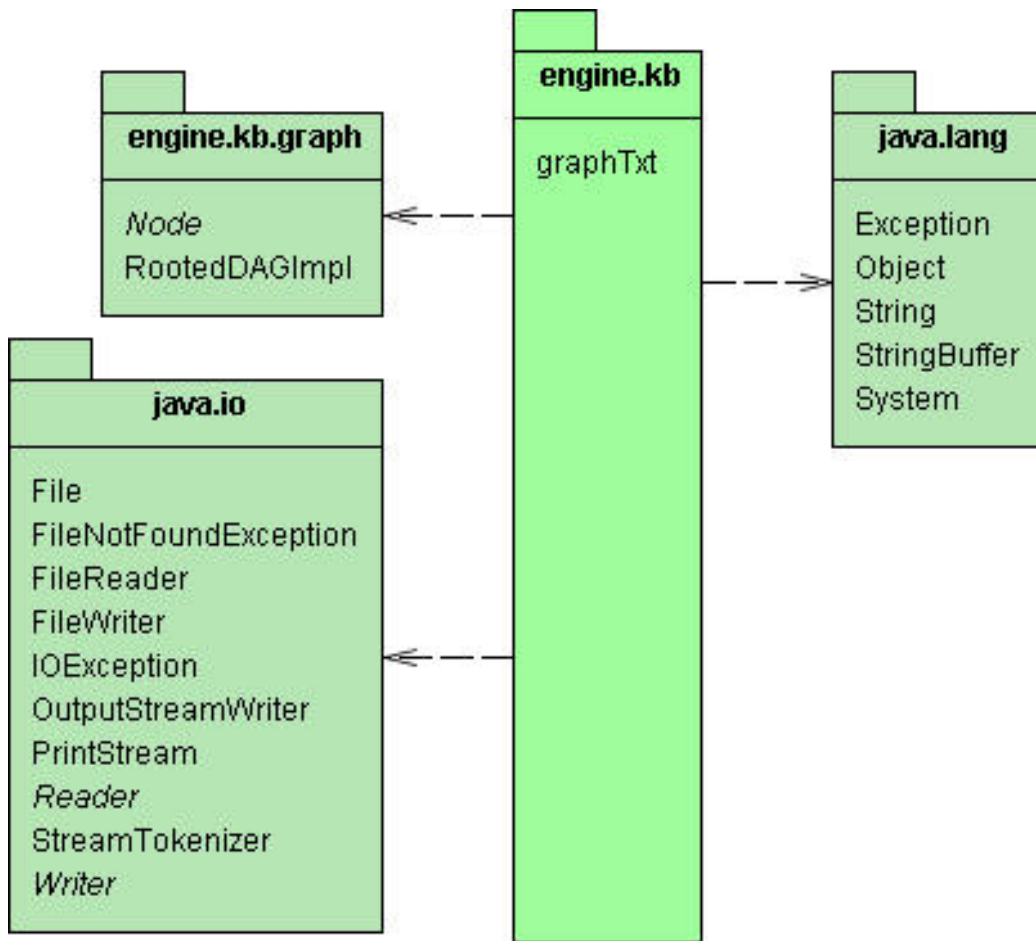


Figure A.2: Package engine.kb Class Diagram

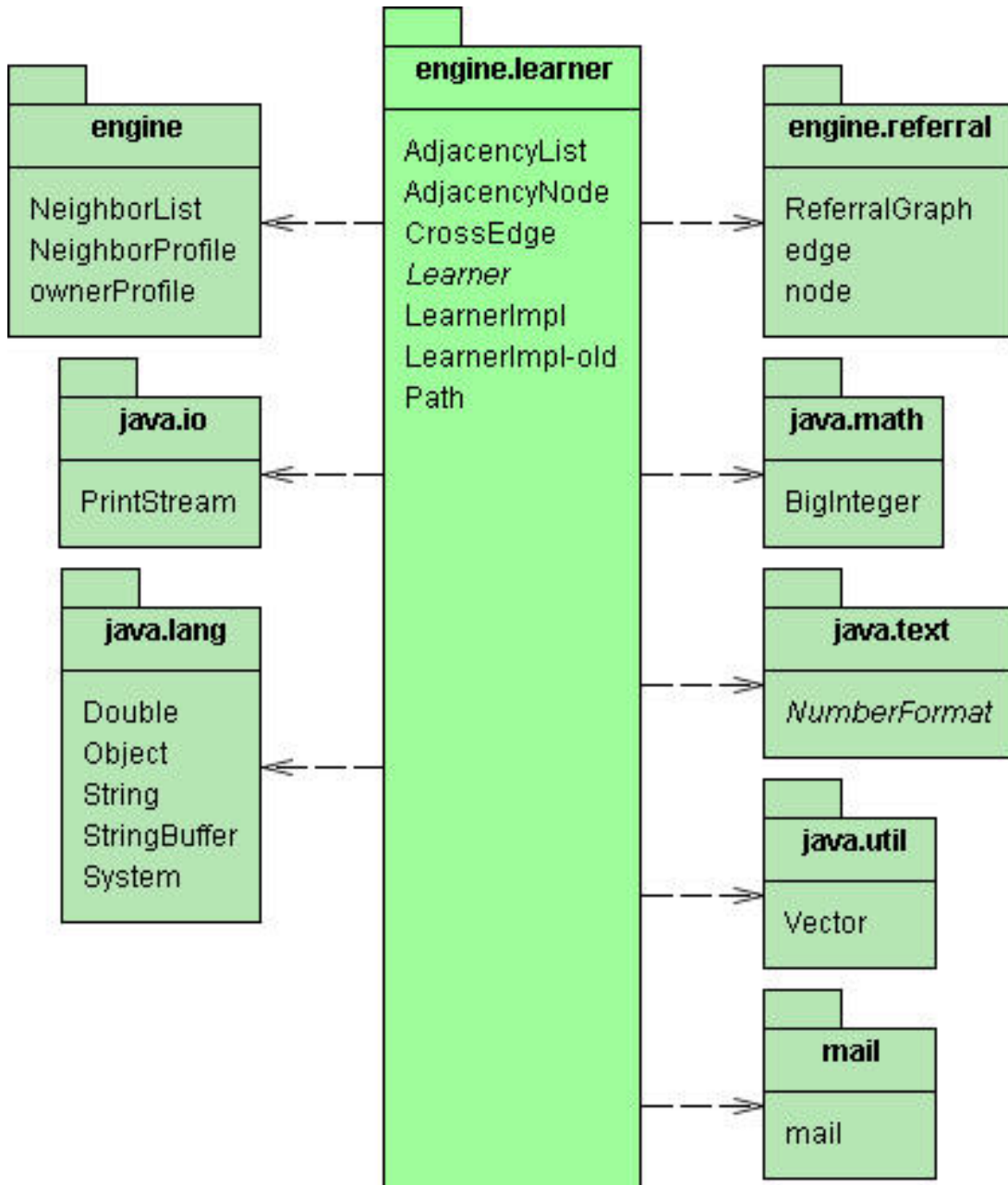


Figure A.3: Package engine.learner Class Diagram

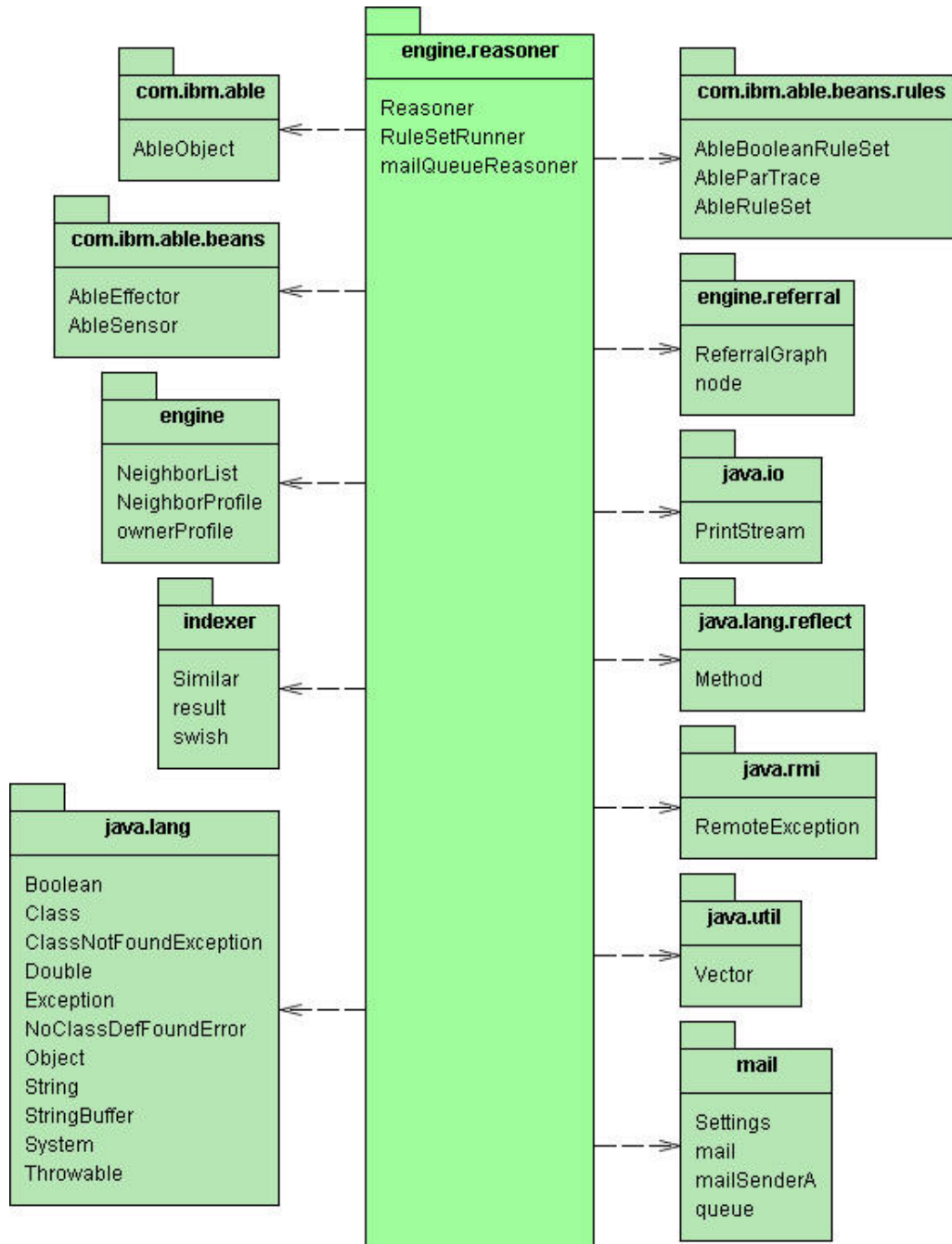


Figure A.4: Package engine.reasoner Class Diagram

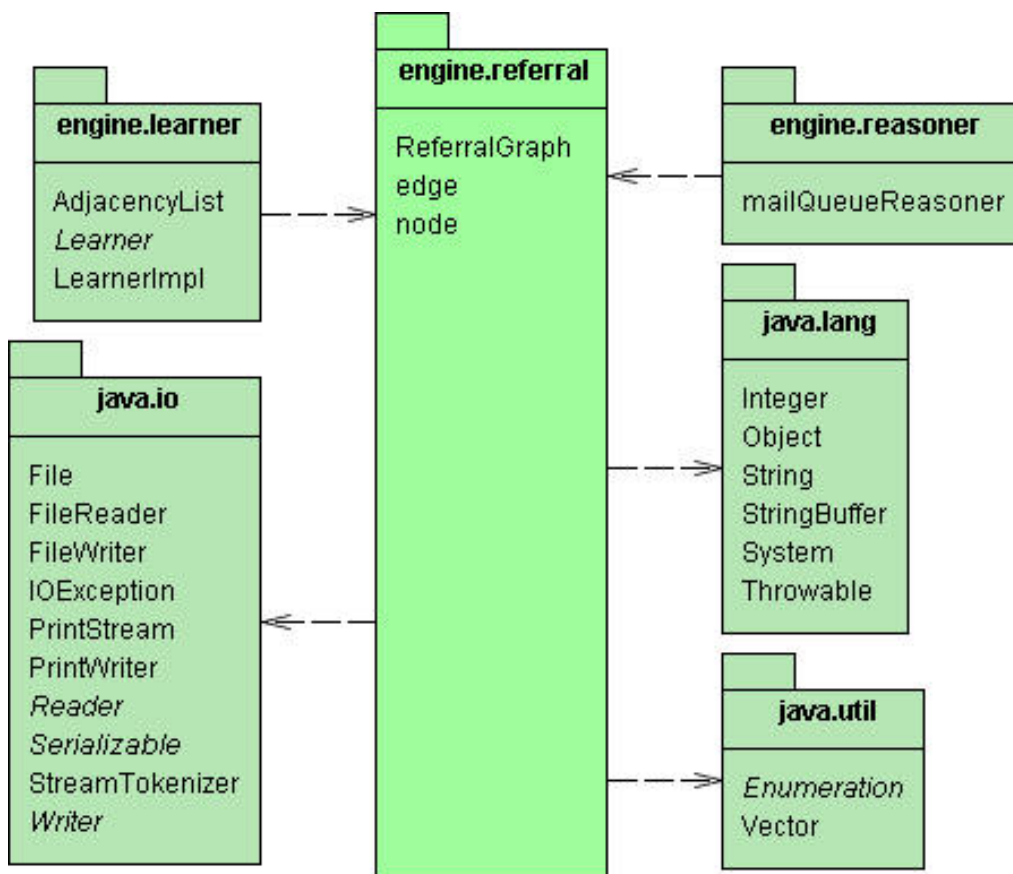


Figure A.5: Package engine.referral Class Diagram

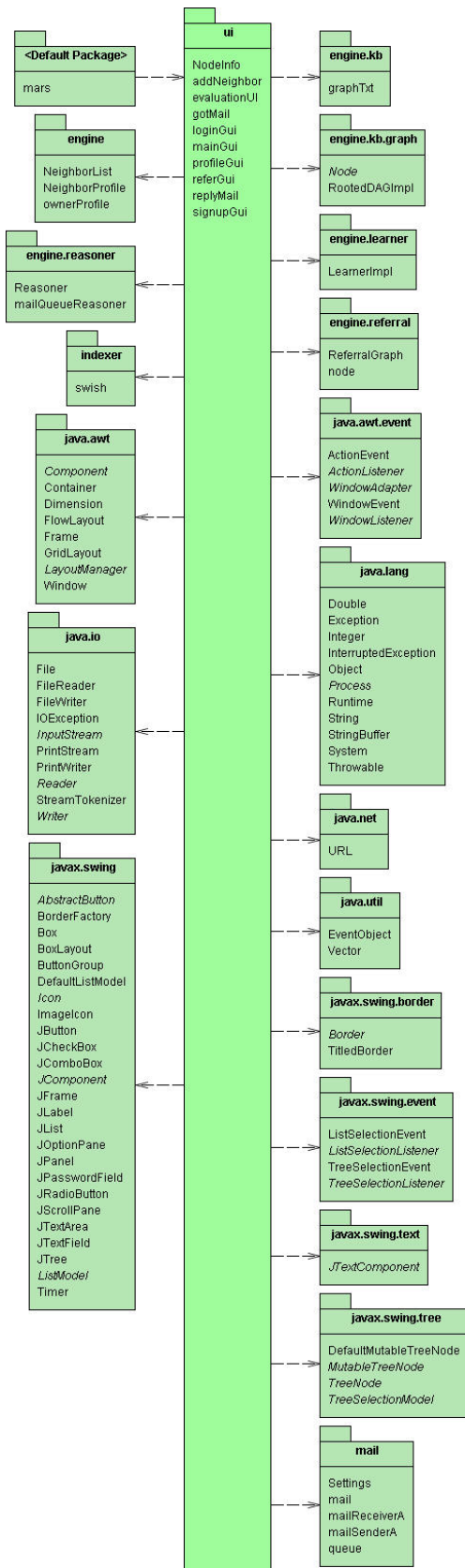


Figure A.6: Package ui Class Diagram

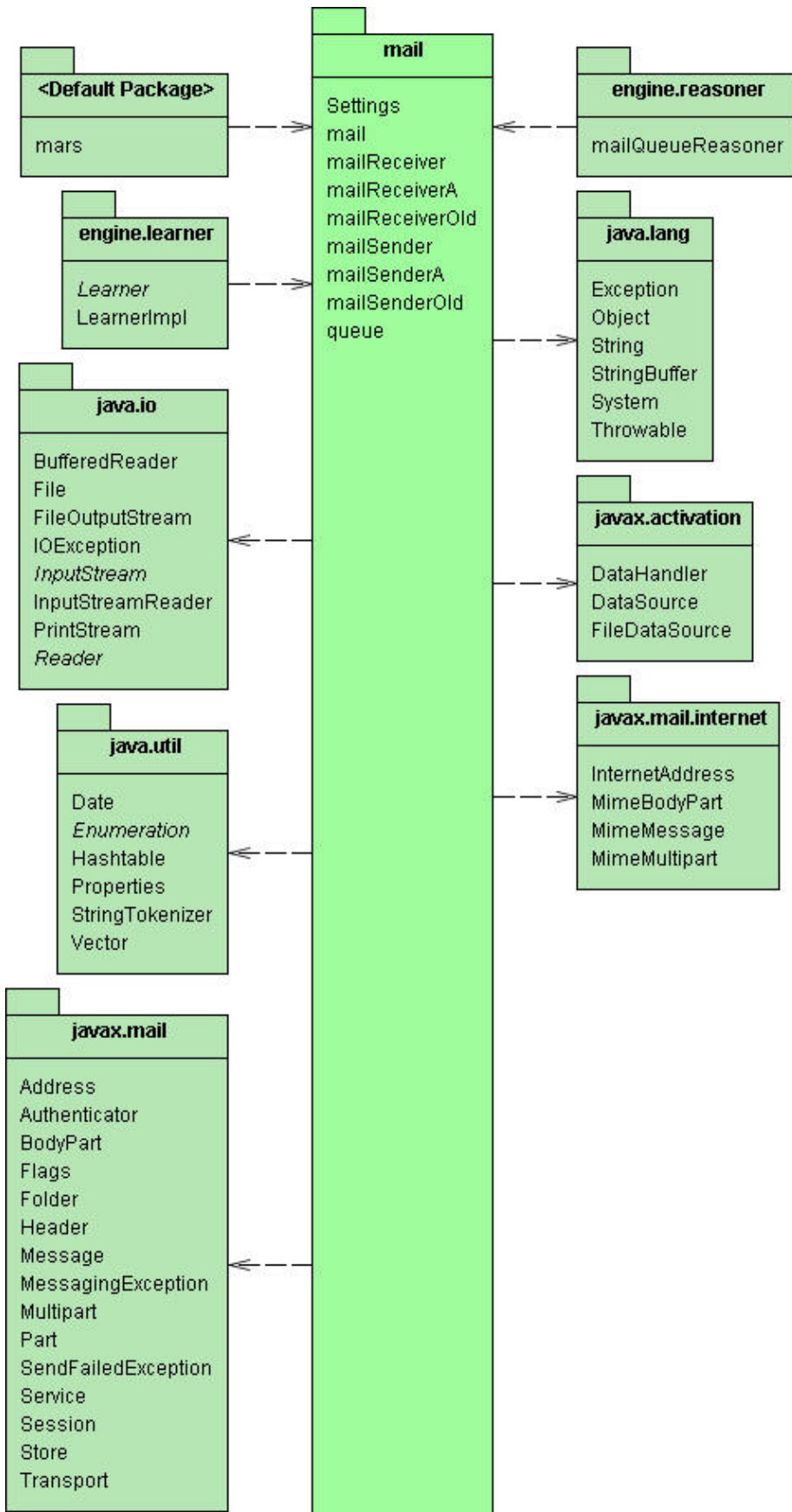


Figure A.7: Package mail Class Diagram

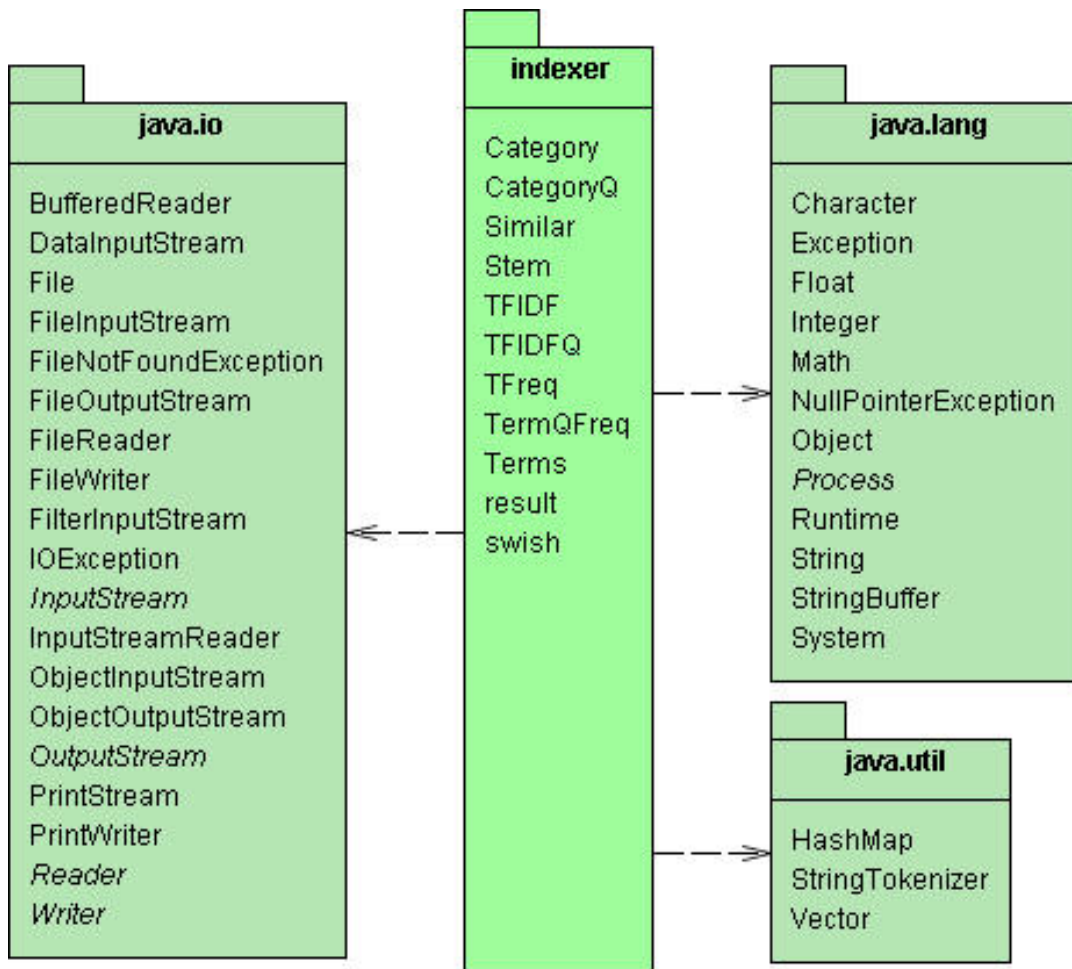


Figure A.8: Package indexer Class Diagram

Bibliography

- Scott L. Alexander and Nancy J. McCracken. Parallel term indexing for a document retrieval system. *A NPAC(National Parallel Architectures Center) REU Project*, 1994.
- Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM review*, pages 573–593, 1995.
- Eric W. Brown, James P. Callan, and W. Bruce Croft. Fast incremental indexing for full-text information retrieval. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB)*, 1994.
- Thomas Dietinger, Christian Gutl, Hermann Maurer, and Maja Pivec. Targeted information retrieval. In *Proceedings of ICCE99*, volume 2, pages 355–358, 1999.
- Oystein Grovlen. Natural language processing in information retrieval. Technical report, Norwegian Institute of Technology, 1995.
- Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.
- Babita Gupta, Lakshmi S. Iyer, and Jay E. Aronson. Knowledge management:practices and challenges. *industrial management and data systems*, 2000.
- Joseph M. Hellerstein, Elias Koutsoupias, and Christos H. Papadimitriou. On the analysis of indexing schemes. In *16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 249–256, 1997.

- Henry Kautz, Bart Selman, and Mehul Shah. Referralweb: Combining social networks and collaborative filtering. In *Communications of the ACM* 40(3), pages 63–65, 1997.
- Todd A. Letsche and Michael W. Berry. Large-scale information retrieval with latent semantic indexing. *Information Sciences*, 100:105–137, 1997.
- Michael Polanyi. *Personal Knowledge: towards a post-critical philosophy*. University of chicago press, 1958.
- Gerard Salton and Chris Buckley. Parallel text search methods. *Communications of the ACM*, 31:203–215, 1988.
- Munindar P. Singh. Degrees of separation. *IEEE internet Computing*, 3:4–5, 1999.
- Tak W. Yan and Hector Garcia-Molina. Index structures for information filtering under the vector space model. Technical report, Department of Computer Science, Stanford University, Stanford CA 94305, 1993.
- Bin Yu and Munindar P. Singh. A social mechanism of reputation management in electronic communities. In *Proceedings of Fourth International Workshop on Cooperative Information Agents*, pages 154–165, 2000.
- Bin Yu, Mahadevan Venkatraman, and Munindar P. Singh. A multiagent referral system for expertise location. In *Working Notes of the AAAI Workshop on Intelligent Information Systems*, pages 66–69, 1999.