MIA
A Multi-Agent Location Based Information Systems for Mobile Users in 3G Networks

Gerd Beuster, Thomas Kleemann, Bernd Thomas
Universität Koblenz-Landau
Instiut für Informatik
{gb|tomkl|bthomas@uni-koblenz.de}

February 25, 2003

The evolving communication infrastructure of the third generation networks provides new possibilities as well as new challenges for information agents. Detailing the architecture of MIA, a prototype of a multi agent information system for mobile users, we describe the impact of upcoming network and client improvements on information agents. The expansion of accessible sources enables the agent system to provide personalized market information just when it becomes available. Due to its autonomous activity the user will not be forced to wait for results after the specification of his wishes. Valuable information services like this will be a driving force in the acceptance of the new mobile network infrastructure.

1 Introduction

With the emerging consumer needs for information accessibility wherever and whenever he desires new challenges for various IT sectors arise. This ranges from the very low level of hardware development for mobile uses, up to high level system designs of autonomous software agents fulfilling individual negotiation and trading tasks for the user. A huge market of location based services is growing. Service portals for mobile users incorporate automatic user location estimation via devices like a mobile phone or GPS system integrated into his car. Older concepts like E-Commerce seem to be overtaken by new ideas like Mobile Marketing, where the mobile user is contacted by marketing agents via SMS services or e-mails whenever he enters a geographical region in which the acquisition of new customers makes sense. Thinking in terms of B2C

acquisition we will yield better results in marketing activity if the information will be preprocessed by a personalized agent. Thus we avoid the waste of communication bandwidth within the network as well as within the acceptance of marketing information by a user. In addition to B2C we also have to consider Consumer2Consumer (C2C) services that can be incorporated into location aware services. The information agent will search thousands of classified ads and will reject those offerings that are out of the users reach. In order to supply a user with local information relevant to her interests her position must be known by the service. This data is provided by mobile phone cell information or by GPS receivers. This idea differs strongly from those services offered by some telecom providers that are statically location based lacking any individualization. Since our approach enables the user to freely define what he is interested in, this will overcome the shortfalls of current services. The major source of information will be the World Wide Web. Containing information to almost every topic the WWW is unstructured and heterogenous. Thus combining such existing services of retrieving its information from manually maintained databases with innovative techniques of automatic information retrieval and extraction methods seems to be a challenging and profitable idea offering the mobile user a surplus.

The incorporation of additional information sources is well covered by the modular architecture of MIA [Beuster *et al.*, 2000b; Beuster *et al.*, 2000a]. Taking into account the improving abilities of the mobile devices and infrastructure the information can be streamed to the mobile device including text and pictures like maps or photos of the desired location and service. This significant advantage of 3G mobile networks enables the MIA system to deliver the same information to mobile users, that is currently limited to the HTML interface. The listing of nearby cinemas and their schedule may be enriched with description of the pictures including photographic material. The use of intelligent information services will be a significant improvement over existing cellular networks.

2 The MIA System

MIA (Mobile Information Agent) is a prototype developed at the University of Koblenz by the Artificial Intelligence Research Group. MIA's main intention is to offer a location based information service for mobile users equipped with a mobile phone or a PDA with online capabilities. In addition to the mobile variant MIA also supports access via a standard web browser (HTML). The prototype version supports an automatic user location estimation via the use of a GPS and PDA.

2.1 Domain Independent Search Profiles

The key feature of MIA is its ability to gather and extract information from web pages according to freely defined topics. A user can define several search profiles, where each profile consists of a number of search topics (e.g. restaurant) and additional restricting expressions (e.g. except

Figure 2: Web Session: Search Profile Definition

italian). Profiles can be changed at anytime, even while the system is gathering information. The system adapts to the all changes immediately. The results reflect the new user profile.

2.2 Automatic User Location Estimation

MIA also supports the automatic modification of its search tasks according to changes of the users location. Therefore it monitors the users position with the help of the users GPS. This monitoring of the GPS data and the automatic notification of position changes are sent by the MIA client application, hosted on the users device, to the MIA system. The use of a geographical database resolve the longitude and latitude coordinates into city names. This city name in combination with the users profile is used for the online search of information. The small footprint of network cells in the 3G mobile phone system will provide a replacement for the GPS system that will be sufficiently accurate for localization based services. Mapping the cell information to the name of the city or suburb will be accomplished by database requests. In addition to the obvious classification of cell information to suburbs, cities, countries etc. we will introduce an individual classification like home, office, shopping. These extension can be handled by the ontologie agent.

2.3 Knowledge Guided Search

Since it is obvious that an online search solely based on the keywords from the user profile will not show good results ontological knowledge is incorporated into the search process. For experimental issues two ontologies based on a description logic system have been implemented.

One ontology covers very frequent concepts from the catering area and the other models very common concepts from the area of recreation. Ontologies formalized in description logics are in general separated into two parts: the terminological part (concept and role definitions) and the assertional part con-taining the instances of concepts and relations. For the reasoning on these structures the KR-Hyper system is used. This combination allows the MIA system to derive similar or related expressions to enrich the profiles entered by the user without bothering the user with addi-tional questions during the profile creation phase. For example, if the user searches for italian restaurants it is reasonable that the MIA systems also takes into consideration web pages that are linked via a link description like ”best pizza” since pizza is a meal offered in an italian restaurant. This is exactly the functionality provided by the KR-Hyper system given the appropriate ontology. The reasoner is capable to derive from the ontology that an italian restaurant offers certain meals and that one of these meals is pizza. Expansion of the knowledge is restricted to changes in the declaration of the ontology and will not affect the system itself. Because the ontology will never cover all aspects of life the information gathering will still be operational except the refinements.

2.4 Anytime Information Access

Based on the fact that online search and extraction depends on network issues, query response times of http servers and by itself is more complex than an ordi-nary database lookup, because more than one online source is examined (in fact the gathering process spiders to many web sites) the overall query response time might be to high. Hence it makes more sense not to wait until all information gathering processes have finished instead these ”spider agents” continously update the search results as soon as they find a new piece of information. The key idea is that of anytime algorithms, since by the huge nature of the web we might run into the problem, that it takes a very long time to visit all relevant web pages by the spider agent. Therefore it is much more reasonable to assign something like a lifetime to each of these agents. If this is expired the agent stops its work, sends its information to a central storage and then terminates. This ensures that no information is lost and the overall system query response time can be decreased for same future search tasks. In the context of mobile users and speaking in terms of online costs and reachability anytime algorithms have a further advantage. After a user has initiated the search he can logout of the system and can relogin at any time he wants to. All tasks started before are still active or after its lifetime ended their results are pre-served. Thus his mobile device is not blocked by the MIA system and the user is not forced to stay online which would produce costs.

2.5 Integration with other Services

Two key components of the MIA system offer a wide flexibility to be integrated or coupled with other services. The information extraction system used within MIA is applicable to any kind of semi-structured documents (HTML,XML, etc.). This component can be used to learn extraction procedures (Wrappers) in a very convenient way. By presenting a small number of

Figure 4: MIA search results on a HTML device

text parts from sample documents the system is able to synthesize programs which later on can be used for automatic extraction tasks applicable to similar documents. The extractions provided by wrappers are in general relational (i.e. extractions are data tuples and each slot has a fixed semantics) and can easily be used to populate databases or may function as input to other services. In the MIA system for example the extracted address information is used to query an online map service for a related map displaying the location of the previously found place of interest. Secondly the ontology reasoning system (KR-Hyper [Baumgartner *et al.*, 2002] plus ontologies) is applicable whenever a reasoning about terminological knowledge and instances is needed. Since neither the system nor the ontology modeling language is domain bound the application of the system to other domains is possible. Furthermore the reasoner can be used as a theorem prover or as a general deductive component.

3 AI Techniques

Within the MIA-system several core technolgies from the field of artificial intelligence are applied. These techniques range from standard methods to special modified and developed methods to provide powerful solutions for the various problems of : web search, text classification, information extraction and deductive reasoning.

3.1 Web Search

Besides connecting various yellow pages online services to the MIA-system, one essential information source is the use of spidering techniques to find appropriate web documents. The first step of the information retrieval process is to find web pages containing information about a special topic. In contrast to existing search engines, our approach searches for relevant web pages

Figure 5: MIA search results linked with Online Map Services

online [Wolff, 2002]. Thus we call this approach spidering, instead of indexing web documents due to estimated word frequencies and building indexed databases. We determine the relevancy of a web document by its context, given by its relation to other pages. Our approach is as follows: we interpret the WWW as a directed graph where web pages are nodes and hyperlinks are vertices labeled with a URL and an associated text. Assume we are given some keywords and some start pages (entry points). A relevant web page is found due to the keywords if, starting

at one of the start pages we can find a path with nodes and vertices labeled with the URL and associated text, such that: 1) every vertex label of this path contains at least one keyword 2) every keyword must at least occur once in one of the vertices labels.

This basic idea can be improved as follows: If we are searching for a site containing information about Chinese restaurants in Heidelberg, and we follow a path where the label of the first edge (l1) in this path contains "heidelberg", l2 to l10 contain "restaurant" and l11 contains "chinese", this last link is very unlikely to have still something to do with "heidelberg", although "heidelberg" occurred somewhere on this path. Therefore we introduce a parameter "maximum keyword distance" defining a context radius for the given keywords. In a similar way, we cut off the search paths if we depart too far from the last occurrence of a keyword: If a keyword has not occurred in the last KN labels in our path, we will not follow this path any further. This parameter is called the "kickout number". Good paths can be found with a modified best first search strategy by using the mentioned keyword distance as the parameter for a cost function similar to standard informed search algorithms. By looking at the according paths of each link to decide which link of a set of links is the one nearest to a possible goal and chose the link whose path has the minimal keyword distance.

3.2 Text Classification

Adding a text classification component [Beuster, 2001] improves both the effectiveness and the efficiency of the system. About 96% of the web pages provided by our spider algorithm do not contain addresses. If we can filter out (parts of) these irrelevant web pages, we can increase the speed of the system significantly. The efficiency is improved, because the information extraction component gets some more information which it can facilitate in its task to extract information from a web page. So far, our information extraction agent can extract addresses from a web page. For this, it has a variety of methods available. Some methods are rather "strict" (extraction sometimes fails, although address information is available), some are rather "loose" (wrong positives on pages that do not contain addresses). With the additional information from the classifier, the information extraction component can make a better decision which method to use. If the classifier gives a high confidence in the quality of the web page, the information extraction agent will also try "loose" methods when the "strict" methods fail. If it is not likely that a web page contains addresses, the information extraction agent will not use "loose" methods but simply abandon the page. We use Artificial Neural Networks for text classification. The network architecture we have chosen is a fully connected back-propagation feed-forward network with 100 input nodes, 50 hidden nodes, and 2 output nodes. We map the web pages onto the input nodes in the following way: With each of a selection of 100 words, an input node is associated. If a web page contains one of these words, the corresponding input node has 1-activation; otherwise, it has 0-activation. The output nodes represent the classification "web page contains an address" and "web page does not contain an address". In the training phase, the appropriate output node is set to 1, and the other output node is set to 0. In the recognition phase, the document is put in the class whose associated output node has the higher activation. The neural network

is trained offline on 3000 pre-classified pages. These pages are authentic web pages gathered by the spider algorithm. The network has been trained until the error did no longer decline on an independent set of another 3000 pre-classified pages. After the training, 98% of the pages from the training set are classified correctly. This excellent classification result is partly due to the distribution of web pages: About 95% of the web pages that are gathered by the spider agent do not contain addresses. On the positive examples, the one that contain addresses, we get a recognition rate of about 60%. The 100 words that are used to give a representation of a web page have been chosen by selecting those 100 words from all the words appearing in the web pages from the training set which contain the most information, i.e. the words that allow best to distinguish between pages that contain address, and pages that do not contain addresses.

3.3 Information Extraction and Machine Learning

To extract information [Kushmerick and Thomas, 2002] from web pages it is either necessary to understand the document, which would involve semantic text analyzation techniques, or to find relevant information by recognizing its underlying syntactical representation. Because linguistically motivated attempts containing semantical and syntactical parsing fail or are at least not very robust on web documents (HTML documents), it makes more sense to use the special text formatting and annotating strings (tags) of these documents to recognize and extract relevant information. One heuristic used in the MIA system is that to use a set of pre-selected web sites

Figure 6: MIA's Wrapper Learning System

as entry points for the retrieval. This assures to the user that at least some results are presented. While some known information resources are used, the problem remains of extracting this information. The MIA administrator uses MIA's wrapper toolkit to learn wrappers (extraction procedures) for certain web domains. Whenever the extraction agents visit one of these domains during their search they use these pre-learned wrappers to extract information from one of the web pages. Currently the wrapper toolkit uses a one shot learning strategy [Grieser *et al.*, 2000; Thomas, 2000; Thomas, 1999a; Thomas, 1999b] extended with a special document representation based on the Document Object Model (DOM) representation. The general strategy to learn left and right anchors to define the start and end point for extraction (delimiters) is kept, but extended to path learning in the DOM of the document. Additionally the general intention is now to learn one wrapper for a whole document class (eg found at one web domain) instead learning one wrapper for one document. This is in contrast to [Thomas, 1999b] and the second method used by MIA. The major problem someone is confronted with in the context of an autonomous

Figure 7: MIA-WLS: Testing a Wrapper

multiagent system like MIA is the lack of available examples for learning wrappers online. Because MIA's web page classifier provides unknown web pages to the system, we can not assume the existence of the training data needed for learning. On the one hand, the classifier is good

enough to determine if an address is contained on a web page; on the other hand, arbitrary web pages vary too much for a single general purpose address wrapper to be effective. While the use of a large address database and a name recognizer (POS-tagger or named-entity recognizer) might work better for this particular problem, but MIA is aiming at a generic approach with no hard-wired solutions.

To overcome this problem MIA uses a learning algorithm that derives its examples by means of knowledge representation techniques. That is, we model our knowledge about addresses with a logic KR language in advance and are able to query this knowledge base to derive example patterns. These example patterns can then be used to construct examples as input to a modified learner. This allows MIA to learn wrappers even for unknown pages. Various experiments showed that the knowledge base can also be used to verify the extracted information on a still limited level, but nevertheless this idea can serve as basis for some kind of self-supervision for autonomous information extraction agents.

3.4 Ontological Knowledge Reasoning

Besides using deductive reasoning processes to enrich the search capabilites of the spider agents (see Section 2.3 Knowledge Guided Search) they are also used for the semantic evaluation of extraction results. Since by the heuristic nature and the theortical boundaries of learning wrappers from positive examples only, it cannot be expected that the extraction results are 100 % correct. Thus each spider agent needs some sort of self control concerning the extractions he did. One very restrictive way to test for correct extractions, is to check if all of the keywords entered by the user are contained in the extraction result. Obviously this restricts the result set to a great extent. A more intelligent way is to check if the content of some of the extraction slots are similar with respect to a given ontology and the user given keywords. This method of extraction evaluation is carried out in the MIA-System by the use of the previously mentioned ontology and the reasoning system KR-Hyper [Baumgartner *et al.*, 2002]. These deductive based evaluation techniques are additionaly supported by cross checking methods for city names and their according ZIP-codes. This is easily established by the use of an ZIP-code database. Filtering the extractions by combining standard database cross checks and with deductive reasoning based evaluation techniques results in very good final results.

3.5 Multi Agent System

Distributing agents over multiple nodes serves two purposes: Increasing the robustness of the system and improving system performance. The system performance of a distributed system depends on the ratio between the computation performed by each agent and the communication between the agents. Until the time needed for communication between agents outweighs the time saved by running agents in parallel, distributing agents speeds up the MAS.

MIA's multi-agent structure comes close to an optimal distribution, because agents communicate very little with each other, and the computational tasks of most of the agents are completely

independent from the tasks of most of the other agents. Communication between agents happens in four situations: on system startup, when a user profile is created or changed, when search agents are started and when results are returned by the search agent. For now, we just want to point out that in each of these cases, the amount of data communicated is very limited: Assigning a search task to an agent involves just two data items: The location of the user, and the search part. The search part con-sists of two keywords, the topic of the feature and the kind of relationship between topic and feature (‘only’ or ‘not’).

Whenever a search agent has found new information, the new information is transferred to the server agent. The information returned by the search agents are (real world) addresses. They consist of four short strings: The name of the venue, its street address, its city and its telephone number. Such a message is usually smaller than 255 bytes. A search agent has to go to several steps in order to get this information: It has to spider the web for promising web pages, classify whether the web page does indeed contain address information, extract the address information, and cross-check whether the extraction was accurate. All of these steps are independent from the activities of the other search agents. Therefore, MIA has very good parallelization and distribution capabilities.

Figure 8: MIA’s Multi Agent Architecture

The design of distributed multi-agent systems usually abstracts from the actual machines running the agents by the concept of agent platforms. An agent platform is an environment for a

number of agents which is visible to the outer world. Other agents can query an agent platform for the agents running on the platform, and can request the start or stop of agents on the platform. There is not necessarily a 1:1 relationship between agent platforms and machines. An agent platform can manage agents hosted on a number of machines, and one machine can host multiple agent platforms. Although not complying completely, MIA's agent system and communication architecture is oriented at FIPA, the most advanced agent standard.

In MIA, the agent platform looks like an ordinary agent to the outer world. This means, all queries and commands to the agent platform are phrased as ordinary KQML-performatives. The only difference between the agent platform and an ordinary agent is that the agent platform can create new agents, and that it keeps a record of the agents created. For this reason, we will call the agent platform the "platform agent" in the following. In the standard case, there is one platform agent running on each of the available machines. Although this is the standard scenario, it is also possible to run multiple platform agents on the same machine. Four other types of agents are interacting within MIA:

Matchmaker Agent: The matchmaker agent is the central broker agent in MIA. Whenever some agent needs assistance in order to fulfill a task, it asks the matchmaker to recommend a suitable agent. The matchmaker agent takes care of getting the agent in touch with a suitable agent. To do so, it asks one of the platform agents to create an appropriate agent. The matchmaker agent knows about the status of all other agents. Whenever an agent's status changes, the information is passed to the matchmaker agent.

Figure 9: Agent communication (search request)

Server Agent: The server agent provides the front-end to the users. It can provide information in HTML, WML format and in a proprietary format used by the MIA-PalmOS-application. The server agent stores user profiles and search results and it is in charge of starting spidering agents when requested by the user.

Blackboard Agent: This agent is a helper for the server agent. The server agent keeps the search results (provided by the spider agents) in memory only. When-ever the system is stopped and restarted the server agent losses the search results. In order to keep them over multiple sessions, the blackboard agent is used. Like the server agent, the blackboard agent subscribes to all results of spider agents. The blackboard agent stores all results in a file. When the system is restarted, the server agent consults the blackboard agent for the results gathered in previous sessions.

Spider Agent: Spider agents do the actual information gathering and extraction. For each search topic of each user and every city the user enters, one spider agent is started. This agent searches for information about the given topic in this city and transfers the results back to the server agent.

Ontology Agent: The ontology agent is used to query the ontology reasoner with various predefined queries. Thus it deals as sort of container for the knowledge representation system.

The agent communication is best illustrated in brief by some sample sessions that take place between agents in order to show the interaction.

Startup: Agent startup is divided in three phases: In the first phase, a platform agent is started on each of the hosts used by the MIA system. In the second phase, on one of the platforms the core agents server, matchmaker, and blackboard are started. In the third phase, the blackboard agent transfers results previously gathered to the server agent. Finally, the blackboard agent subscribes to agent status information from the server agent. This way, whenever the spider agent gets in contact with a new (spider) agent, the blackboard agent knows about, and subscribes to the search results of the spider agent.

Start Search: Before the spiders start to gather information from the WWW, two preliminaries must be fulfilled: The user who wants to use the system must have created a search profile with the server agent, and she must have sent her current position. When she submits her position, the following communication takes place for each of the search topics form her profile: The server agent, who is in need of a spider agent, asks the matchmaker agent to recommend a spider agent to use. The matchmaker's strategy in this situation is to create a new spider agent. Therefore, it sends a "create spider" request to its platform agent (PlatformB). When the spider is created, it registers with PlatformB. From there, the information about the new spider goes to the matchmaker, which forwards it to the server. The server issues the search request to the spider, and subscribes to its results. The spider transfers its new status "starting" to the the server. Since the blackboard agent gets informed by the matchmaker agent about all agent status changes, it notices the new spider and subscribes to its results as well. Since MIA works asynchronously, the spider does not send back an answer immediately, but continuously collects results. Therefore, the server has to send a second command to the spider, explicitly telling it that all results gathered should be forwarded to the spider.

Collecting results: Since both the spider and the server have subscribed to all new results found by one of the spider agents, each new information is forwarded to these two agents as soon as a spider agent finds it. The user gets to see the search results when she requests them

from the server.

As mentioned before, in order to maximize the utilization of available computational resources, every distributed multi-agent system has to maximize two goals: a) Distribute the computations among the machines available for best utilization of the computational resources b) Minimize the communication between the ma-chines. MIA was designed to run on uniform off-the-shelf computers connected in an in-house network, as is typical the case in universities and research institutes. Especially during start-up, there is a lot of communication between the blackboard agent and the matchmaker agent, but neither the blackboard agent nor the matchmaker agent have intense computational tasks to fulfill. On the other hand, in normal MIA operation, the search agents do need a lot of computational power for information classification and extraction, but they communicate only little with other agents. Therefore, we can use a simple agent distribution strategy: On the first agent platform the core agents matchmaker, server, and blackboard are started. If more than one platform is available, the spider agents are equally distributed over all platforms.

Under the assumption that all machines have the same computational power avail-able, and the network connection to the machine running the server agent is equally well for all machines, this guarantees an optimal distribution of the agents. MIA does not attempt to improve the load balancing on its own. If MIA is running in heterogeneous networks with machines of varying power, the user can do manual load-balancing by starting multiple platform agents on powerful machines. Since new agents are distributed equally between the agent platforms, this ensures that more powerful machines run more agents.

4 Conclusion and Future Work

The architecture of MIA has proved to be extensible and flexible. The incorpora-tion of the KR-Hyper system to reason about the ontologies boosted the quality of the information search well beyond the results of available search engines. The analysis and recompilation of the information gathered is an essential step towards mobile information systems. The mobile phone infrastucture like UMTS in Germany, that is currently under implementation, enables our system to provide full access to individually selected content overcoming the limitation of widely used GSM communication tech-niques. Parallel to these enhancements, the 3G network will allow the MIA-clients to incorporate localization services without additional equipment like GPS receivers. The availability of easily usable but powerful information agents is part of the content that has to be developed for 3G networks in order to make the new gen-eration a success. MIA and 3G provide enhancements to each other. The use of cellular phones as the client device enables the service to be reached almost everywhere, and enables charging for the informations. These charging ca-pabilities open a wide area of informations. We plan to integrate search agents for classified ads. Currently these ads are sold by regional papers. The combination of ontology based search and localization services offers new opportunities for the client, because he will be informed about new offers independent of his ability to access and search the paper himself.

Earnings may be split among the publisher and network operators. The customer may benefit from the savings in time and he will not miss an issue, because his agent won't.

References

[Baumgartner *et al.*, 2002] Peter Baumgartner, Ulrich Furbach, and Bernd Thomas. Model based deduction for knowledge representation. In *Proceedings of the 17. WLP - Workshop Logische Programmierung*, September 2002.

[Beuster *et al.*, 2000a] Gerd Beuster, Bernd Thomas, and Christian Wolff. MIA - A Ubiquitous Multi-Agent Web Information System. In *Proceedings of International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000)*, December 2000.

[Beuster *et al.*, 2000b] Gerd Beuster, Bernd Thomas, and Christian Wolff. Ubiquitous Web Information Agents. In *Proceedings of Workshop on Artificial Intelligence In Mobile Systems*, August 2000. European Conference on Aritifical Intelligence (ECAI).

[Beuster, 2001] Gerd Beuster. MIC - A System for Classification of Structured and Unstructured Texts, March 2001. Diploma Thesis.

[Grieser *et al.*, 2000] Gunter Grieser, Klaus P. Jantke, Steffen Lange, and Bernd Thomas. A Unifying Approach to HTML Wrapper Representation and Learning. In *Proceedings of the Third International Conference on Discovery Science*, December 2000. Kyoto, Japan.

[Kushmerick and Thomas, 2002] Nicholas Kushmerick and Bernd Thomas. *Intelligent Information Agents R&D in Europe: An AgentLink perspective*, chapter Adaptive Information Extraction: A Core Technology for Information Agents. Springer, 2002.

[Thomas, 1999a] Bernd Thomas. Anti-Unification Based Learning of T-Wrappers for Information Extraction. In *Proc. AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[Thomas, 1999b] Bernd Thomas. Learning T-Wrappers for Information Extraction. In *Workshop on Machine Learning in Human Language Technology*, July 1999. ACAI'99 Advanced Course on Artificial Intelligence.

[Thomas, 2000] Bernd Thomas. Token-Templates and Logic Programs for Intelligent Web Search. *Intelligent Information Systems*, 14(2/3):241–261, March-June 2000. Special Issue: Methodologies for Intelligent Information Systems.

[Wolff, 2002] Christian Wolff. Web spidering with ai methods, 2002. Studienarbeit, Universität Koblenz-Landau, Institut für Informatik.