# Ontology-Focused Crawling of Web Documents

Marc Ehrig
Institute AIFB
University of Karslruhe
Karlsruhe, Germany
ehrig@aifb.uni-karlsruhe.de

Alexander Maedche
Knowledge Management Department
FZI Research Center for Information Technologies
Karlsruhe, Germany
maedche@fzi.de

## ABSTRACT

The Web, the largest unstructured database of the world, has greatly improved access to documents. However, documents on the Web are largely disorganized. Due to the distributed nature of the World Wide Web it is difficult to use it as a tool for information and knowledge management. Therefore, users doing the difficult task of exploring the Web have to be supported by intelligent means.

This paper proposes an approach for document discovery building on a comprehensive framework for ontology-focused crawling of Web documents. Our framework includes means for using a complex ontology and associated instance elements. It defines several relevance computation strategies and provides an empirical evaluation which has shown promising results.

Keywords: Web Searching and Crawling, Semantic Web

## 1. INTRODUCTION

The enormous growth of the World Wide Web in recent years has made it important to develop document discovery mechanisms based on intelligent techniques such as **focused crawling** [3, 6, 1]. In its classical sense a crawler is a program that retrieves Web pages. This service is commonly used by search engines or Web caches to build up their repositories. Crawling is the basic technique for building huge data storages. Focused crawling goes a step further than the classic approach. It is able to crawl particular topical (focused) portions of the World Wide Web quickly. It is therefore very attractive for scenarios where not the whole Web is of interest, but only a certain domain. These can be specialized search engines or companies trying to gain information in their field of activity. A focused crawling algorithm loads a page and extracts the links. By rating the links based on keywords the crawler decides which page to retrieve next. Link by link the Web is traversed. Our crawling framework builds on and extends existing work in the area of focused document crawling. We do not only use keywords for the crawl, but rely on high-level background knowledge with concepts and relations, which are compared with the texts of the searched page. This way we can easily achieve a direct focus. Crawling the Semantic Web with actual metadata will further reveal the power of our approach. Here, we propose an ontology-focused crawling framework and its implementation CATYRPEL, that provide the following main contributions: An ontology structure extended for the purposes of the focused crawler, several new and innovative approaches for relevance computation based on conceptual and linguistic means reflecting the underlying ontology structures, a comprehensive tool environment which supports both the management of the focused crawling process and the management of the ontology, and an empirical evaluation which shows that crawling based on background knowledge clearly outperforms standard focused-crawling techniques.

*SAC 2003,* Melbourne, Florida, USA
Copyright 2003 ACM 1-58113-624-2/03/03 ...$5.00.

## 2. AN ONTOLOGY-FOCUSED CRAWLING FRAMEWORK

The focused crawling process consists of two interconnected cycles: First, the ontology cycle, and second, the crawling cycle (Fig. 1). The first cycle is mainly driven by the human engineer. He defines the crawling target in the form of an instantiated ontology. This cycle also provides the output of the crawling process to the user in the form of a document list and proposals for enhancement of the already existing ontology to the user. The second

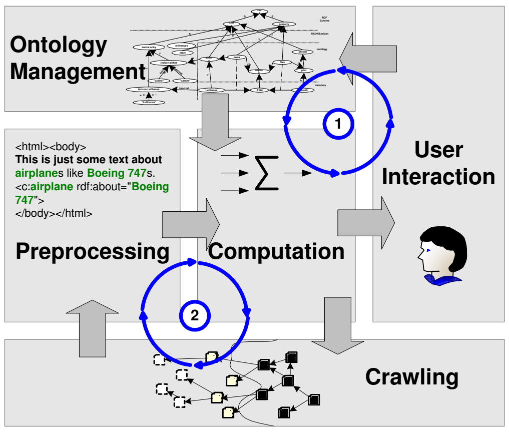


**Figure 1: The Focused Crawling Process and its relation to the architectural components**

cycle comprises the Internet crawler. It interacts automatically with the data contained on the Web and retrieves them. Then it connects to the ontology to determine relevance. The relevance computation is used to choose relevant documents for the user and to focus on links for the further search for relevant documents and metadata available on the Web.

The two connected cycles serve as input for the specification of the system architecture, where we pursue a modular, component-based approach. It roughly consists of the following **five core com-**

**ponents**: *(i)* User Involvement and Interaction, *(ii)* Ontology Management, *(iii)* Web Crawling, *(iv)* Preprocessing, and *(v)* Relevance Computation:

*User Interaction.* The human user provides input to the crawling process along several dimensions: The initial ontology is the background knowledge the crawler relies on. Often the ontology already exists in the structure of a company's knowledge management system e.g. in the form of thesauri and libraries like the ACM classification system. The user can adjust the relevance computation strategy, if he has special crawling needs[1]. A set of start URLs has to be defined by the user. Within and after the crawling process the user has to deal with the different outputs of the crawler: A document list with an assignment to the top ten entities (most relevant entities contained in the document), and a maintenance module, which offers frequent lexical entries and relationships between lexical entries. Users should add these terms to their ontology to improve it.

*Web Crawling.* The Web crawler is started with a given set of links. The links are retrieved in the order of their rank (normally assigned by the relevance measure which will be introduced later). The Web documents are then passed on to the preprocessing module. To ensure a smooth retrieval a buffer of links, several parallel retrieval threads and a Web document buffer are utilized. Links and documents have to be checked to prevent doubles due to redirected links or mirror sites.

*Preprocessing.* Preprocessing is splitted into several steps. **Shallow text preprocessing techniques** for normalization of the free text are applied. In our current implementation we use the well-known porter stemming algorithm [11]. The modular approach of the architecture also allows us to use more sophisticated linguistic processing techniques such as provided by the GATE[5] system for English or the SMES[10] system for German.

*Ontology Management.* The crawler bases its focus by using the provided graph as background knowledge for its search in the Web. We support the user in **engineering** the ontology by providing graphical means and several consistency ensuring as well as verification techniques.

*Relevance Computation.* This component is a core element of the overall approach. In general the relevance measure is a function which tries to map the content (e.g. natural language text, hyperlinks, etc.) of a Web document against the existing ontology to gain an overall **relevance-score**. The measures and the associated relevance computation strategies are described in detail in section 4.

## 3. ONTOLOGIES, METADATA AND A LEXICAL LAYER

It has been widely accepted that ontologies and metadata are the core elements for the Semantic Web. These will have to be extended for the relevance measures of our focused crawler. In the following we introduce a formal model of our notion of ontologies and metadata, where a specific focus is set on the interaction of ontology and associated metadata with natural language (KAON)[9].

DEFINITION 1 (ONTOLOGICAL LAYER). *Ontology $\mathcal{O} := \{\mathcal{C}, \mathcal{P}, \mathcal{H}^{\mathcal{C},\mathcal{P}}, prop\}$, with **classes** $\mathcal{C}$, **properties** $\mathcal{P}$, a **class hierarchy** $\mathcal{H}^{\mathcal{C}}$: $\mathcal{H}^{\mathcal{C}} \subseteq \mathcal{C} \times \mathcal{C}$ ($\mathcal{H}^{\mathcal{C}}(C_1, C_2)$ means that $C_1$ is a sub-class of $C_2$), a **relation** function $prop : \mathcal{P} \to \mathcal{C} \times \mathcal{C}$ (Domain dom: $\mathcal{P} \to \mathcal{C}$ with $dom(P) := \Pi_1(prop(P))$, and range: $\mathcal{P} \to \mathcal{C}$ with $range(P) := \Pi_2(prop(P)$; for $prop(P) = (C_1, C_2)$ one may also write $P(C_1, C_2)$)), and a **property hierarchy** $\mathcal{H}^{\mathcal{P}}$: defined analogously to $\mathcal{H}^{\mathcal{C}}$[2].*

DEFINITION 2 (METADATA LAYER). *Metadata structure $\mathcal{MD} := \{\mathcal{O}, \mathcal{I}, inst, instr\}$, with ontology $\mathcal{O}$, **instances** $\mathcal{I}$, a **class instantiation** function $inst : \mathcal{C} \to 2^{\mathcal{I}}$ (for $inst(C) = I$ one may also write $C(I)$), and a **property instantiation** function $instr : \mathcal{P} \to 2^{\mathcal{I} \times \mathcal{I}}$ analogously to inst.*

As the crawling process typically operates on natural language documents, the core layers presented above are augmented with a lexical layer that facilitates the linking of documents to ontological entities.

DEFINITION 3 (LEXICAL LAYER). *Lexicon $\mathcal{L} := \{\mathcal{L}^{\mathcal{C}}, \mathcal{L}^{\mathcal{P}}, \mathcal{L}^{\mathcal{I}}, \mathcal{F}, \mathcal{G}, \mathcal{J}\}$, with **lexical entries** $\mathcal{L}^{\mathcal{C}}, \mathcal{L}^{\mathcal{P}}, \mathcal{L}^{\mathcal{I}}$ for $\mathcal{C}, \mathcal{P}, \mathcal{I}$, and **references** $\mathcal{F}$, $\mathcal{G}$, $\mathcal{J}$. Based on $\mathcal{F} \subseteq \mathcal{L}^{\mathcal{C}} \times \mathcal{C}$, let for $L \in \mathcal{L}^{\mathcal{C}}$, $\mathcal{F}(L) = \{C \in \mathcal{C} | (L, C) \in \mathcal{F}\}$ and for $\mathcal{F}^{-1}(C) = \{L \in \mathcal{L}^{\mathcal{C}} | (L, C) \in \mathcal{F}\}$. $\mathcal{G}$ and $\mathcal{J}$ are defined analogously.*

The definition allows n:m-relations between lexical entries and ontological entities, that is a lexical entry may refer to several classes or properties and one class or property may be referenced by several lexical entries.

*An Example.* The ontology as depicted in figure 3 contains several concepts like AIRPLANE, A340, PASSENGER, and instances as MARC EHRIG. TRANSPORTS represents a relation. An example for the lexical layer of AIRPLANE would be {"Airplane","Aeroplane","Plane"}. The meaning of the circles will be explained in the next section.

## 4. RELEVANCE COMPUTATION

The most important component within the crawling framework is the relevance computation component. This section gives a detailed overview on the relevance computing strategy as it is used within our framework.

DEFINITION 4 (BASIC PREMISES). *Instantiated ontology plus lexicon $\mathcal{O}_{\mathcal{L}}$, entities $E$ ($E = C \cup I \cup P$), and a subset being relevant for the user's focused crawling process $E^U \subseteq E$.*

In general the relevance measure is a function which tries to map the content (e.g. natural language text, hyperlinks, etc.) of a Web document against a ontology to gain a relevance-score.

DEFINITION 5 (RELEVANCE FUNCTION). *Relevance score $r := f(d, \mathcal{O}'_{\mathcal{L}})$, with $r \in \boldsymbol{R}$, the input document d, and the instantiated ontology $\mathcal{O}'_{\mathcal{L}}$.*

The mapping from a given document $d$ to a final relevance-score is achieved in a **three step process** that is applied to the preprocessed document: Establishing Entity Reference, Background Knowledge Compilation, and Score Summarization.

[1]An explanation of what base strategies are best is given in the evaluation part.

[2]The reader may note, that this generic definition follows RDF, which does not distinguish between associations and attributes.

*An Example for the Relevance Computation Process.* We motivate and introduce the relevance computation process by a simple example (Fig. 2). The example shows how the relevance is calculated from a given Web document. The relevance is calculated in several steps passing from left to right. It starts with word stemming and word counts, is followed by considerations about relations, and finishes with a summarization function. This example results in an overall relevance-score of $4$, based on the usage of relational relevance sets and the sum function that will be described in detail below.

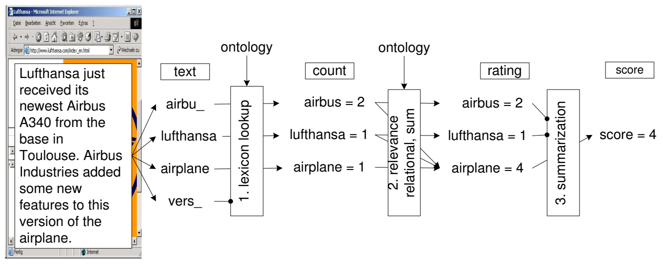

Lufthansa just received its newest Airbus A340 from the base in Toulouse. Airbus Industries added some new features to this version of the airplane.

**Figure 2: Different Relevance Computation Steps**

In the following we give a detailed introduction for the three steps of the overall relevance computation process.

*Step I: Establishing Entity Reference.* Free text has already been adjusted using the preprocessing module. This step is a lookup of entities in the lexicon of the ontology. Every time a word matches an entry in the lexicon, and therefore an entity of the ontology structure, the specific entity is counted. The results are qualified by multiplying them with the well known tf/idf weights provided by the information retrieval community (see [13]) and dividing them by the text length.

DEFINITION 6 (ENTITY LOOKUP). *Frequency $f_d^{\mathcal{L}}(e) = k_e$ of entity $e$ in document $d$ based on lexical entry lookup in the document from lexicon $\mathcal{L}$.*

*Step II: Background Knowledge Compilation.* The next step uses the structure of the graph to assign relevance scores to single entities. The computation takes into account the relevance scores of entities which are ontological related. For each entity a set of related entities is defined. The sum of scores of the entities in the set generates the score of the original entity. The function resembles a filter which lets the scores of some entities pass (especially entities being closely related to the goal entity) and prevents others from influencing the outcome.

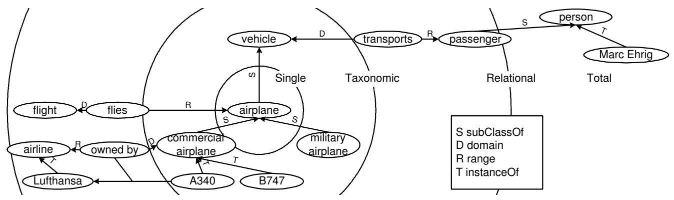


**Figure 3: Relevance Sets**

DEFINITION 7 (RELEVANCE SETS). *We distinguish between **Single** $R_s(e)$, **Taxonomic** $R_t(e)$, **Relational** $R_r(e)$, and **Total** $R_o^n(e)$ relevance sets. The weight for each entity in the sets is generally 100%.*

*$R_s(e)$ is the most simple measure returning nothing more than just the original list of entities referred to in a document, viz. $R_s(e) = \{e\}, \forall e \in E$.*

*For $R_t(e)$ the list of related entities is defined by the entity itself and the entities which are directly[3] connected to it following the taxonomical tree aka. a distance measure of one in the ontology viz. $R_t(e) = R_s(e) \bigcup \forall e_t \in E, d(e, e_t) = 1$.*

*$R_r(e)$ considers the ontology graph. The list of related entities is not constrained to taxonomic relations. Additional entities are appended: all directly linked entities, including relations and their ranges aka. a distance measure of two, viz. $R_r(e) = R_t(e) \bigcup \forall e_r \in E, d(e, e_r) = 2$.*

*$R_o^n(e)$ takes the whole ontology as input, viz. $R_o^n(e) = E$. The difference between the entities in the set is the actual weight. The weight is defined as $w_{e_s}$ with $n$ being the discount factor and $d(e_s, e)$ being defined as the distance between an entity of the set $e_s$ and the core entity $e$, viz. $w_{e_s} = 100\% \times (n\%)^{d(e_s,e)}$. Any other discount function could also be chosen, e.g. linear or logarithmic functions. The only restriction is to give entities which are more distant a lower weight than the nearer ones.*

Figure 3 depicts the different relevance sets using the simple example introduced before. The figure is to be read from the center to the outside. Using the simple Single relevance measure $R_s$(AIRPLANE) only lexical entries referring to the core entity AIRPLANE are considered. The Taxonomic relevance measure $R_t$(AIRPLANE) includes super- and sub-concepts like VEHICLE, COMMERCIAL AIRPLANE and MILITARY AIRPLANE. The complex Relational relevance measure $R_r$(AIRPLANE) also includes non-taxonomic relations like TRANSPORTS, FLIES, and OWNED BY. Their corresponding entities are also included. The widest measure called Total $R_o^{50}$(AIRPLANE) also includes entities being far off the original entity like PERSON and MARC EHRIG. However their weight diminishes with a factor of 50% for each distance step.

To summarize the scores of the related entities within one set the following approach was chosen:

DEFINITION 8 (ENTITY SCORE). *Add up the marks of each set entity, viz. score $r(e) = \sum_{\forall e_s \in R(e)} k_{e_s} \times w_{e_s}$, with $r(e)$ being the relevance score of entity $e$, $R(e)$ representing one of the four defined relevance sets $R_s(e), R_t(e), R_r(e), R_o^n(e)$, $e_s$ the entity within the relevance set, $w_{e_s}$ the weight, and $k_{e_s}$ the result of the entity lookup obtained by step I.*

At the end of the second step we receive a new list of the entities represented in the document with their respective scores.

*Step III: Summarization.* The final step is used to summarize only the entities a user has initially defined in his search space $E^U$. Only the scores of the chosen entities will be rated for the final relevance of the document. They are summarized and a single figure is gained.

DEFINITION 9 (SUMMARIZATION). *$r$ is defined as:*

- *$r = \sum r(e)$ with $e \in E^U$.*
- *$r = min(r(e))$. This reflects the* `AND`*-conjunction.*
- *$r = max(r(e))$. This reflects the* `OR`*-disjunction.*

With this scalar rating $r$ the user can choose among his ranked documents. Additionally the crawler can optimize its further search by following highly ranked links first.

[3]Only directly related entities are used, as a use of all inherited entities would not represent the original goal entities any longer. The level of generalization would be too broad.

## 5. EVALUATION AGAINST BASELINES

The evaluation study shows how the different relevance measures perform in real life and how the quality of the input ontology influences the performance of the crawler. We compared the different relevance computation strategies (taxonomic, relational, and total) introduced in section 4 with **two baseline crawling approaches**, namely standard breadth-first search crawling and focused crawling with simple keyword spotting. We use three scenarios based on three knowledge bases.

*Relevance Computation Strategies / Baselines.* Standard breadth-first takes the given initial document list and processes them using breadth-first search. The keyword-based approach evaluates a given Web page using a set of predefined keywords (in our case just taken from the lexicon). If one of the predefined keywords appears on the Web page, the page is considered as relevant, if not it is skipped. The taxonomic, relational, and total relevance strategies work with the defined set of entities using the available background knowledge. All three use a simple sum function (as described in the last section) to compute the overall relevance-score.

*Scenarios.* We here provide a description of the three different crawling scenarios, each with one of the data sets, we have performed within our empirical evaluation study.

The first scenario uses the **university domain**. Goal for this crawling scenario is to find all pages about the Center of Intelligent Information Retrieval, an institute at the University of Massachusetts (CIIR_AMHERST_UMASS). Direct lexical entries for this entity are "intelligent information" and "CIIR". As starting URL "www.umass.edu" was selected and only this domain was allowed for search. The knowledge representation in form of an ontology for this scenario comprises of 76 nodes and 224 lexical entries referring to these nodes.

The second scenario is a commercial one. To find documents about the Boeing 747 the crawler accessed the **airplane ontology**. The focus has been set on the entity BOEING_747 and its lexical entries "boeing 747" and "747". To make this run more interesting the "boeing.com" domain was blocked. The crawler started off from "www.umass.edu", "www.cnn.com", "www.lufthansa.com", "www.ual.com", "www.travel.com", and "www.massachusetts.com". Here the ontology included 51 nodes and 170 lexical entries.

The final **tourism scenario** looks for pages covering hotel and waterfront. Hotels on the sea or on the lake border are the goal, a typical scenario when somebody plans a vacation. Keywords are "hotel" and "waterfront". The lexical entries are a bit wider ("motel", "inn", "resort", "oceanside", "beachfront"). The start URLs are to follow: "www.travel.org", "www.hotel.com", "www.guest.com", "www.nbc.com", and "www.byebye.com". The tourism scenario has 52 nodes and 167 lexical entries.

*Metric.* Perhaps the most crucial evaluation of focused crawling is to measure the rate at which relevant pages are acquired, and how effectively irrelevant pages are filtered off from the crawl. The metric used in our scenario is the well-known **harvest rate** [3, 1].

$hr = \frac{\#r}{\#p}, hr \in [0, 1]$.

It represents the fraction of Web pages crawled satisfying the crawling target $\#r$ among the retrieved pages $\#p$. This harvest ratio must be high, otherwise the focused crawler would spend a lot of time merely eliminating irrelevant pages, and it may be better to use an ordinary crawler instead. Thus, a high harvest rate is a sign of a good crawling run.

For evaluating our focused crawler we defined a "satisfying crawling target" as a page which contains the keywords of the searched concept. This is not a constraint to evaluating the system, because every relevance strategy faces this same restricting definition of the crawling target. The strategies can therefore still be easily compared.

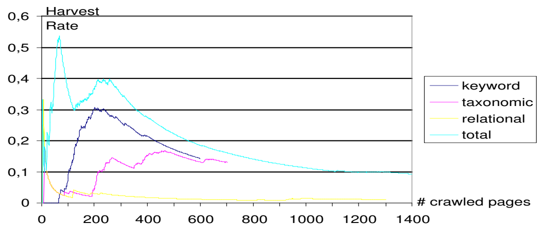


**Figure 4: Evaluation in scenario 1**

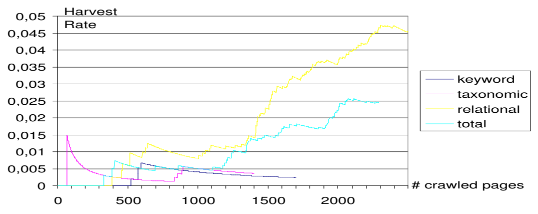


**Figure 5: Evaluation in scenario 2**

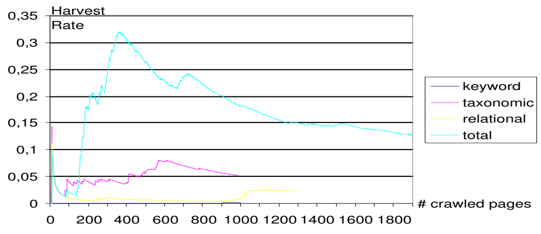


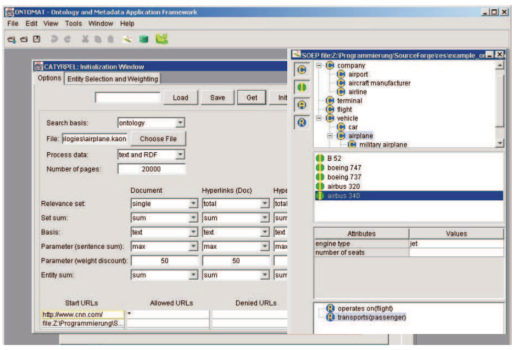
**Figure 6: Evaluation in scenario 3**

*Experimental Results.* The described scenarios have been processed several times to ensure the quality of the results. Representative for all the crawling scenarios the results of the last scenario are described in the following paragraph (Figure 6). The other runs are presented only graphically.

Breadth-first is identically to the x-axis. No relevant pages were found by this base line technique. Keyword found two pages, not a very good result either. The taxonomic relevance strategy shows the first considerable results with relational being a little bit lower. The best strategy is represented by total. The other scenarios have similar results as one can see in the plots. Using the whole ontology as background clearly outperforms the standard crawling approaches. Using our focused crawling approach achieves better results than the techniques used today.

## 6. THE IMPLEMENTATION: CATYRPEL

CATYRPEL, the implementation of our document crawling framework, is an application within the KAON, the Karlsruhe Ontology and Semantic Web tool suite [9]. CATYRPEL is embedded into the ONTOMAT-Ontology and Metadata Application Framework (see [8]), which is a comprehensive and flexible integrated


ONTOMAT - Ontology and Metadata Application Framework
File Edit View Tools Window Help
CATYRPEL: Initialization Window
Options
Entity Selection and Weighting
Load
Save
Get
Init
Choose File
SOEP file:Z:\Programmierung\SourceForge\res\example_on

Search basis: ontology
File: ologies\airplane.kaon
Process data: text and RDF
Number of pages: 20000

| | Document | Hyperlinks (Doc) | Hype |
|---|---|---|---|
| Relevance set: | single | total | total |
| Set sum: | sum | sum | sum |
| Basis: | text | text | text |
| Parameter (sentence sum): | max | max | max |
| Parameter (weight discount): | 50 | 50 | |
| Entity sum: | sum | sum | sum |

| Start URLs | Allowed URLs | Denied URLs |
|---|---|---|
| http://www.cnn.com/ | * | |
| file:Z:\Programmierung\S... | | |

company
airport
aircraft manufacturer
airline
terminal
flight
vehicle
car
airplane
military airplane

B 52
boeing 747
boeing 737
airbus 320
airbus 340

| Attributes | Values |
|---|---|
| engine type | jet |
| number of seats | |

operates on(flight)
transports(passenger)

**Figure 7: The CATYRPEL environment**

development environment. The underlying data structure is provided by KAON-API. Figure 7 depicts a screenshot of the running CATYRPEL system. On the left side the crawling starter interface is shown, where all the described initializations can be made. The tight integration of the crawler with the ontology and metadata management component is also important to allow for quick adaption and extension of the structures.

## 7. RELATED WORK

The idea of combining focused crawling with a complex knowledge representation as in ontologies is new and to the best of our knowledge until now it has not been pursued in research. However, there is several work around that shares similarities with our approach. [3] present a generic architecture of a focused crawler. The crawler uses a set of predefined documents associated with topics in a Yahoo like taxonomy to build a focused crawler. [7] focus on measuring the user interest. Additionally they come up with some further ideas for a generic focused crawler. Some interesting aspects about when to terminate a focused crawl are also mentioned. However, no detailed evaluation of their approach is provided. The idea of tunnelling [2] through several links without actually rating them is a proposal on how to prevent getting caught in one domain. Whereas our idea is based on text and contents, the approach presented in [6] uses so-called context graphs as a means to model the paths leading to relevant web pages. Context graphs in their sense represent link hierarchies within which relevant Web pages occur together. A combination of context graphs and ontology-focused crawling might be future work. [4] did experiments with different crawling strategies (keyword comparisons, link metrics). [12] propose a machine learning oriented approach for focused crawling. Their crawler uses reinforcement learning to learn to choose the next link in a way that over time a reward is maximized. A focused crawler which tries to learn the linkage structure while crawling is described in [1]. This involves looking for specific features in a page which makes it more likely that it links to a given topic. These features may include page content, URL structure, link structure (siblings etc.). The authors argument that the learning approach is a more general framework than assuming a standard pre-defined structure while crawling.

## 8. CONCLUSION

In this paper we have introduced a new approach for focused crawling based on a graph-oriented knowledge representation. The framework included a definition of the knowledge structure and its lexical presentment, relevance computation strategies, an implementation, and an empirical evaluation of the overall framework. Our approach goes beyond existing focused crawling, because it consequently uses the ontology as background for focusing the search in the Web. We have provided a quality-based empirical evaluation based on a standard metric harvest-rate. The study has shown that our approach clearly outperforms typical breadth-first, keyword- or taxonomic-based approaches. It can therefore be regarded as a tool for efficient knowledge sharing in the World Wide Web.

In the future we will have to evaluate our crawler in a larger application scenario including document and metadata discovery, as the future Semantic Web will have. We are currently working on an application of our crawler for discovering Web services, in our case news provided in the RSS news syndication format[4] and extended with a domain specific ontology. Thus, in the future, discovery of complex Web service may be approached using our crawling approach.

## 9. ACKNOWLEDGEMENTS

The research presented in this paper would not have been possible without our colleagues, at the Institute AIFB, University of Karlsruhe, and FZI, Karlsruhe. We thank L. Stojanovic, R. Volz, and S. Handschuh, who did major work on defining and implementing the framework, on which we built or work on.

## 10. REFERENCES

[1] C. C. Aggarwal, F. Al-Garawi, and P. Yu. Intelligent crawling on the world wide web with arbitrary predicates. In *WWW-10, Hong Kong*, 2001.

[2] D. Bergmark, C. Lagoze, and A. Sbityakov. Focused crawls, tunneling, and digital libraries. In *ACM European Conference on Digital Libraries*, Rome, September 2002.

[3] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. In *WWW-8*, 1999.

[4] J. Cho, H. García-Molina, and L. Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.

[5] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, July 2002.

[6] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused Crawling using Context Graphs. In *VLDB-00, 2000*, 2000.

[7] M. Ester and M. Gross. Ariadne: a focused crawler with adaptive classification of the hyperlinks. In *Nat. Symp. on Machine Learning (FGML '2000), Birlinghoven*, 2000.

[8] S. Handschuh, A. Maedche, and S. Staab. CREAM — Creating relational metadata with a component-based, ontology driven framework. In *SWWS'01, Stanford, USA*, August 2001.

[9] S. Handschuh, A. Maedche, L. Stojanovic, and R. Volz. KAON - The KArlsruhe ONtology and Semantic Web Infrastructure. Technical report, Forschungszentrum Informatik Karlsruhe, 2001. http://kaon.semanticweb.org.

[10] G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An information extraction core system for real world german text processing. In *ANLP-97*, Washington, USA, 1997.

[11] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[12] J. Rennie and A. McCallum. Using Reinforcement Learning to Spider the Web Efficiently. In *ICML-99*, 1999.

[13] G. Salton. *Automatic Text Processing*. Add.-Wesley, 1988.

[4]www.purl.org/rss/1.0/schema.rdf