Ontology-Focused Crawling of Web Documents and RDF-based Metadata

A. Maedche and M. Ehrig and S. Handschuh and L. Stojanovic and R. Volz

1 Forschungszentrum FZI, Research Group WIM, 76131 Karlsruhe, Germany
   http://www.fzi.de

2 Institute AIFB, University of Karlsruhe, Germany
   http://www.aifb.uni-karlsruhe.de

Abstract. The enormous growth of the World Wide Web in recent years has made it important to develop document discovery mechanisms based on intelligent and focused crawling techniques. The next-generation Web, the Semantic Web, that is currently being developed as a meta Web, building on the existing one, changes the classical crawling task. Metadata that is based on ontologies will exist in the form of distributed fragments over the Web. Thus, means for the intelligent and ontology-focused metadata discovery are required. In this paper we propose a comprehensive framework for ontology-focused crawling of Web documents and RDF-based relational metadata in parallel. Our framework includes means for adding a lexical layer to RDF(S)
-based ontologies and metadata, relevance computation strategies, an implementation, and an empirical evaluation, which has shown promising results.

1 Introduction

Recently, it has been seen that ontologies and metadata are the key ingredients for the Semantic Web. In this context, ontologies describe domain theories for the explicit representation of the semantics of metadata. One of the core design principles and a reason for the success of the WWW was decentralization. The same should apply for the Semantic Web with the result that anyone should be able to design or reuse an existing ontology, to define metadata according to this ontology, and, finally to put this metadata on the Web without any registration process. At this point it becomes obvious that metadata will exist in distributed fragments over the Web. Thus, means for the intelligent and ontology-focused metadata discovery are required.

The enormous growth of the World Wide Web in recent years has made it important to develop document discovery mechanisms based on intelligent and focused crawling techniques. In its classical sense a crawler is a program that retrieves Web pages, commonly used by a search engine or a Web cache. Focused crawling is a technique which is able to crawl particular topical (focused) portions of the World Wide Web quickly without having to explore all Web pages. In this paper we propose means

1 We use “RDF(S)” to refer to the combined technologies of RDF and RDF-Schema.
2 http://www.w3.org/DesignIssues/Principles.html
for the intelligent, ontology-focused discovery of distributed RDF-based metadata and documents in parallel. The crawling framework builds on and extends existing work in the area of intelligent and focused document crawling. We here propose an ontology-focused crawling framework and its implementation CATYRPEL\(^3\), that provide the following main achievements:

- An ontology and metadata management infrastructure, including a lexical layer for RDF(S)-based ontologies and metadata;
- Several new and innovative approaches for relevance computation based on conceptual and linguistic means reflecting the underlying ontology structures;
- A comprehensive tool environment supporting the management of the focused crawling process as well as the ontology and metadata engineering and evolution process;
- Finally, the overall crawling framework has been empirically evaluated. The evaluation results have shown that crawling based on background knowledge using ontologies clearly outperforms standard focused-crawling techniques.

**Organization.** This paper is organized as follows. In section 2 we introduce the overall framework, the underlying process, the architecture and its components. As mentioned above, our crawling framework relies on a specific notion of RDF(S)-based ontologies and metadata, which is presented in section 3. An important aspect of this approach is the strict separation between the new introduced lexical knowledge and classes, properties, or concrete instances describing the ontology and metadata. The proposed model serves as a basis for the relevance computation process described in section 4 that guides the focused search on the Web. The empirical evaluation study analyzing the behaviour of the relevance computation strategies within concrete crawling runs is described in section 5. Section 6 deals with the implementation of CATYRPEL and its embedding into the KAON Karlsruhe Ontology and Semantic Web infrastructure\(^4\). Finally, before we conclude and outline future work, we give an overview on related work on focused document and metadata crawling in section 7.

## 2 An Ontology-Focused Crawling Framework

The focused crawling process that we support consists of two interconnected cycles: First, the ontology and metadata management cycle, second, the crawling cycle. Figure 1 depicts the two cycles. The first cycle is mainly driven by the human engineer. He defines the crawling target in the form of the instantiated ontology and the metadata, respectively. This cycle also provides the output of the crawling process to the user in the form of a document list, further metadata structures, and proposals for ontology evolvement to the user. The second cycle comprises the Internet crawler. This cycle interacts automatically with the data contained on the Web and retrieves them. It then connects to the ontology, the metadata, and their lexicons to determine relevance. The

\(^3\) Middle English for “caterpillar”, a larva of a butterfly crawling around in plants.

\(^4\) [http://kaon.semanticweb.org](http://kaon.semanticweb.org)
relevance is used to choose relevant documents for the user and to focus on links for the further search for ontology-relevant metadata and documents available on the Web.

These two connected cycle 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 components: (i) User Involvement and Interaction, (ii) Ontology and Metadata Management, (iii) Web Crawling, (iv) Preprocessing, and (v) Relevance Computation:

User Interaction. The human user provides input to the crawling process along several dimensions. First, if not available, the user has to provide an initial ontology. Second, the user may develop and instantiate the ontology with concrete metadata, e.g. by connecting to an existing database or via annotation. Third, the user has to select an relevance computation strategy. Fourth, a set of start URL’s has to be defined by the user. Within and after the crawling process the user has to deal with the different output of the crawler. First, he will get the document list with an assignment to the top ten entities (most relevant entities contained in the document). Second, the extracted metadata is shown according to the specified ontology. Third, the maintenance module offers frequent lexical entries and relationships between lexical entries to be added to the ontology.

Web Crawling. The Web crawler is started with a given set of URLs. The URLs are retrieved in the order of their rank (normally assigned by the relevance measure). The web documents are then passed on to the preprocessing module. To ensure a smooth retrieval a URL buffer, several parallel retrieval threads and a web document buffer are provided within our framework. URLs and documents have to be double checked to prevent retrieving the same pages more than once due to redirected links or mirror sites. Formats like PDF and Postscript are converted into text.
Preprocessing. Preprocessing is split into several steps. First, if RDF statements are available in the Web document, they are extracted and copied into a separate file. Second, different shallow text preprocessing techniques for normalization of the free text are applied. In our current implementation we use the well-known porter stemming algorithm [7]. It is important to mention that the modular approach of the architecture also allows us to use more sophisticated linguistic processing techniques such as provided by the GATE\textsuperscript{5} system for English or the SMES\textsuperscript{6} system for German.

Ontology and Metadata Management. As mentioned above the crawler bases its focus by using the given ontology and, if available, existing metadata as background knowledge for its search in the Web. The ontology and metadata management component supports the user in engineering this kind of data. It provides graphical means and includes several consistency ensuring as well as verification techniques. Additionally, it allows to validate the set of crawled metadata instances against the ontology.

Relevance Computation. This component is the “heart” 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 and, if available, the RDF-based metadata contained in a Web document against an ontology and its existing, already collected metadata to gain an overall relevance-score. The measures and the overall 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. 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. To this extend, we have developed a layered architecture (KAON). We here only present the part of our overall ontology and metadata model that is actually used by the crawler.

Definition 1 (Ontological Layer). An ontology structure is a 4-tupel $O := \{C, \mathcal{P}, \mathcal{H}^C, \mathcal{H}^P, \text{rel}\}$, consisting of two disjoint sets $C$ and $\mathcal{P}$ whose elements are called class and property identifiers (URIs), respectively, a class hierarchy $\mathcal{H}^C$: $\mathcal{H}^C$ is a directed relation $\mathcal{H}^C \subseteq C \times C$ which is also called taxonomy. $\mathcal{H}^C(C_1, C_2)$ means that $C_1$ is a sub-class of $C_2$, a function $\text{prop} : \mathcal{P} \rightarrow C \times C$, that relates classes non-taxonomically\textsuperscript{7}. The function $\text{dom}: \mathcal{P} \rightarrow C$ with $\text{dom}(P) := \Pi_1(\text{rel}(P))$ gives the domain of $P$, and range: $\mathcal{P} \rightarrow C$ with $\text{range}(P) := \Pi_2(\text{rel}(P))$ give its range. For $\text{prop}(P) = (C_1, C_2)$ one may also write $P(C_1, C_2)$. The property hierarchy $\mathcal{H}^P: \mathcal{H}^P$ is defined analogously to the class hierarchy. Thus, a directed relation $\mathcal{H}^P \subseteq \mathcal{P} \times \mathcal{P}$ exists, where $\mathcal{H}^P(P_1, P_2)$ means that $P_1$ is a sub-property of $P_2$.

\textsuperscript{5} GATE: General Architecture for Text Engineering, see http://gate.ac.uk/
\textsuperscript{6} www.dfki.de/neumann/pd-smes/pd-smes.html
\textsuperscript{7} The reader may note, that this generic definition follows RDF, which does not distinguish between associations and attributes.
As the crawling process typically operates on natural language documents, the core ontology layer presented above is augmented with a lexical layer that facilitates the linking of documents to ontological entities (that is classes and properties).

Definition 2 (Lexical Layer for the Ontology). A lexicon for the core ontology structure $O := \{C, P, F, G\}$ consisting of two sets $\mathcal{C}$ and $\mathcal{P}$, whose elements are called lexical entries for classes and properties, respectively, and two relations $F \subseteq \mathcal{C} \times \mathcal{C}$ and $G \subseteq \mathcal{P} \times \mathcal{P}$ called references for classes and properties, respectively. Based on $F$, let for $L \in \mathcal{C}$, $F(L) = \{C \in \mathcal{C} | (L, C) \in F\}$ and for $F^{-1}(C) = \{L \in \mathcal{C} | (L, C) \in F\}$. $G$ and $G^{-1}$ 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.

Definition 3 (Metadata Layer). A metadata structure is a 4-tupel $MD := \{O, I, inst, instr\}$, that consists of an ontology $O$, a set $I$ whose elements are called instance identifiers (correspondingly $C$, $P$ and $I$ are disjoint), a function $inst : C \rightarrow 2^I$ called class instantiation (For $inst(C) = I$ one may also write $C(I)$), and a function $instr : P \rightarrow 2^I$ called property instantiation (For $inst(P) = \{I_1, I_2\}$ one may also write $P(I_1, I_2)$).

Definition 4 (Lexicon for the Metadata). A lexicon for the metadata structure $MD := \{O, I, inst, instr\}$ is a tupel $L^{MD} := (\mathcal{C}, F)$ consisting of a set $\mathcal{C}$ whose elements are called lexical entries for instances, respectively, and a relation...
$\mathcal{J} \subseteq \mathcal{L}^I \times \mathcal{I}$ reference for instances, respectively. Based on $\mathcal{J}$, let for $L \in \mathcal{L}^I$, $\mathcal{J}(L) = \{I \in \mathcal{I} | (L, I) \in \mathcal{J}\}$ and for $\mathcal{J}^{-1}(I) = \{L \in \mathcal{L}^I | (L, I) \in \mathcal{J}\}$.

An Example. Let us consider a short example of an instantiated ontology and metadata structure as depicted in figure 2. Here $\mathcal{C} := \{\text{FLIGHT, AIRPLANE}\}$, $\mathcal{P} := \{\text{FLIES}\}$, and the property $\text{FLIES(FLIGHT, FLIGHT)}$ with its domain/range restrictions are defined. The lexical layer is given by $\mathcal{L}^C = \{\text{"Airplane"}, \text{"Aircraft"}\}$ and $\mathcal{L}^R = \{\text{"fly"}\}$.

The function $\mathcal{F}$ and $\mathcal{G}$ map the lexical entries to the classes and properties of the ontology. $\mathcal{F}$ is applied as follows: $\mathcal{F}(\text{"Airplane"}) = \text{AIRPLANE}, \mathcal{F}(\text{"Aircraft"}) = \text{AIRPLANE}$ and $\mathcal{G}(\text{"fly"}) = \text{FLIES}$. Additionally, the following RDF-based metadata statements are defined: Assume $\mathcal{I} := \{\text{FLA12URI, B747URI}\}$. $\text{inst}$ is applied as following: $\text{inst}(\text{FLA12URI}) = \text{A:FLIGHT}, \text{inst}(\text{B747URI}) = \text{AIRPLANE}$. Similarly to the lexical entries of classes and properties the lexical entries of metadata instances may have values, e.g. in this example $\mathcal{L}^I := \{\text{"Boeing 747\text{"\}}. \mathcal{J}$ is applied as follows: $\mathcal{J}(\text{"Boeing 747\text{"\}} = \text{B747URI}$.

4 Relevance Computation

As mentioned above one of the most important components within the crawling framework is the relevance computation component. We here provide a detailed overview on the relevance computing strategy that is used within our framework. To simplify the further explanation some abbreviating, basic definitions are required at this point.

Definition 5 (Basic Definitions). An instantiated ontology plus lexicon $\mathcal{O}'_I$ is defined as $\mathcal{O}'_I = \{\mathcal{C}, \mathcal{L}_C\}$. The instantiated metadata structure plus lexicon $\mathcal{M}\mathcal{D}'_I$ is defined as $\mathcal{M}\mathcal{D}'_I = \{\mathcal{M}, \mathcal{L}_{MD}\}$. The set of entities contained in $\mathcal{O}'_I$ and $\mathcal{M}\mathcal{D}'_I$ is defined as $\mathcal{E} = \{\mathcal{C}, \mathcal{P}, \mathcal{I}\}$. Among those entities the user chooses a set being relevant for his focused crawling process $\mathcal{E}^U \subseteq \mathcal{E}$. Let sub-entities be $\text{sub}(e) = \{e_{sub}[\mathcal{H}^C, \mathcal{P}(e_{sub}, e)] | e \in \mathcal{E}\}$. Let super-entities be $\text{super}(e) = \{e_{super}[\mathcal{H}^C, \mathcal{P}(e, e_{super})] | e \in \mathcal{E}\}$. Let the concept of an instance be $\text{con}(e) = \{e_{con}[\text{inst}(e_{con}) = e] | e \in \mathcal{I}\}$. Let the relations of a concept be $\text{rel}(e) = \{e_{rel}[\text{rel}(e_{rel}) = (e, e)] | e, e \in \mathcal{C}, \text{inst}(e), \text{range}(e) \text{ and } \text{dom}(e) \text{ have already been defined in the previous section.}$

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 and, if available, the RDF-based metadata contained in a Web document against an ontology and its existing, already collected metadata to gain a relevance-score.

Definition 6 (Relevance Function). The function $f$ takes as input a document $d$, the instantiated ontology $\mathcal{O}'_I$ and metadata $\mathcal{M}\mathcal{D}'_I$ structures. It results in $r := f(d, \mathcal{O}'_I, \mathcal{M}\mathcal{D}'_I)$, with $r \in \mathcal{R}$ being the relevance score.

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:
1. The first step is an **ontology lookup** of the metadata contained in the document (validation) or a **lexicon lookup** (of $\mathcal{L}_O, \mathcal{L}_{MD}$) for the lexical entries contained in the document. This step results in the set of entities that are referred to by the document.

2. The second step uses the structure of the ontology to compute the **relevance of an entity** in dependence of its relation to other entities. This step results in a score that is computed for each entity contained in the ontology and the metadata, respectively.

3. The final step is used to finally **summarize the entities** a user has initially defined in his search space. This step results in the overall and final relevance score $r$ for a given document $d$.

**An Example for the Relevance Computation Process.** We motivate and introduce the relevance computation process by a simple example. The example shows how the relevance of a given Web document (describing some information in the area of aviation) consisting of free, natural language text and RDF-based metadata. Figure 3 depicts the simple example Web document that includes free text and some RDF statements. The upper track represents the free text relevance processing. The lower track shows the metadata relevance processing. A corresponding instantiated airplane ontology and metadata structure$^8$ are depicted graphically in Figure 4.

---

This example results in an overall relevance-score of 6, based on the usage of relational relevance sets and the sum function that will be described in detail below.

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

---

$^8$ The ontology is available online at http://kaon.semanticweb.org/examples/airplane.kaon.
Step I: Entity Reference. As mentioned above, the first step is a lookup of entities in the ontology or in its lexicon. If RDF-based metadata is embedded in the document, the contained identifiers are used to find matches with the ontology. In case of a positive result the respective entities will be counted. Normal free text has already been preprocessed using the preprocessing module. It is now checked via lookup in the ontology and metadata lexicons. Every time a lexical entry matches an entry in the lexicons, and therefore an entity of the ontology and metadata 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 [9]) and dividing them by the text length.

Definition 7 (Entity Lookup). Let $e \in E : f^{\mathcal{O},\mathcal{M},\mathcal{D}}_d(e) = ki_e$ with $ki_e$ being the frequency of entity $e$ in document $d$ based on the lookup of metadata identifiers contained in the document, based on the instantiated ontology $\mathcal{O}$ and the metadata structure $\mathcal{M},\mathcal{D}$. Let $e \in E : f^{\mathcal{L}_o,\mathcal{L}_m,\mathcal{D}}_d(e) = kl_e$ be the frequency of entity $e$ in document $d$ based on the lookup of lexical entry references contained in the document, based on the lookup of lexical entries contained in the lexicons $\mathcal{L}_o$ and $\mathcal{L}_m,\mathcal{D}$. Either one of $ki_e$ or $kl_e$ can be used with the following functions. Ideally the results are both added up in step III.

Step II: Background Knowledge Compilation. The second step uses the knowledge of the ontology to give single entities a relevance score by checking the scores of other, related, entities as well. It is the core part of the relevance measure. For each entity a set of related entities is defined. The 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.

Definition 8 (Relevance Sets). Four different relevance sets $R(e)$ are defined in the context of the focused crawler. We distinguish between Single $R_s(e)$, Taxonomic $R_t(e)$, Relational $R_r(e)$ and Total $R^\mathcal{T}_s(e)$. The weight for each entity in the sets is 100% if not explicitly defined otherwise.

- $R_s(e)$ is the most simple measure returning nothing more than just the original list of entities referred to in a document, viz. $R_s(e) = \{e\}, \forall e \in E$.
- $R_t(e)$ uses the power of ontologies only to a certain extent. The list of related entities is defined by the entity itself, the direct super-entities and the direct sub-entities. This corresponds to a distance measure of one in the ontology graph, viz. $R_t(e) = R_s(e) \cup \{\text{sup}(e), \text{super}(e), \text{inst}(e)\}, e \in \{C, P\}$.
- $R_r(e)$ uses deeper knowledge from the ontology. The list of related entities is defined like the taxonomic measure but additional entities are appended: directly linked concepts, instances and their relations plus the ranges of the relations. This corresponds to a distance measure of two, viz.

\[ R_r(e) = R_s(e) \cup \{\text{con}(e)\}, e \in \{I\} \]

\[ R^\mathcal{T}_s(e) = R_s(e) \cup \{\text{com}(e)\}, e \in \{T\} \]

\[ R^\mathcal{T}_s(e) = R_s(e) \cup \{\text{com}(e)\}, e \in \{T\} \]

[9] Only directly related entities are used, as a use of all inherited entities would not represent the original entities any longer. The level of generalization would be too broad.
\[ R_s(e) = \bigcup_{e' \in \{ \text{sub}(e), \text{super}(e) \}} R_s(e'), \{ \text{rel}(e') \}, \text{range}(\text{rel}(e')) \}, e \in \{ \mathcal{C} \}, \]
\[ R_p(e) = R_s(e) \cup \{ \text{dom}(e), \text{range}(e) \}, e \in \{ \mathcal{P} \}, \]
\[ R_r(e) = R_s(e) \cup \{ \text{rel} \left( \text{con}(e) \right), \text{range}(\text{rel}(\text{con}(e))) \}, e \in \{ \mathcal{I} \}. \]

- \( R^0_e(e) \) (Total) takes the whole ontology and metadata structure as input, viz. \( R^0_e(e) = E \).

The difference between the entities in the set is the actual weight. In the given environment the weight is defined as \( w_{e_i} \), 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. \( w_{e_i} = 100\% \times (n\%)^d(e_s, e) \).

Any other discount function could also be chosen, of course also e.g. linear or logarithmic functions. The only restriction is to give entities which are more distant a lower weight.

**Fig. 4. Relevance Sets**

Figure 4 depicts the different relevance sets using a simple example. The figure is to be read from the center to the outside. Using the simple Single relevance measure \( R_s(\text{AIRPLANE}) \) only lexical entries referring to the core entity \( \text{AIRPLANE} \) are considered. The Taxonomic relevance measure \( R_t(\text{AIRPLANE}) \) includes super- and sub-concepts like \( \text{VEHICLE}, \text{COMMERCIAL AIRPLANE} \) and \( \text{MILITARY AIRPLANE} \). The complex Relational relevance measure \( R_r(\text{AIRPLANE}) \) also includes non-taxonomic relations like \( \text{TRANSPORTS}, \text{FLIES} \), and \( \text{OWNED BY} \). Their corresponding entities are also included. The widest measure called Total \( R^0(\text{AIRPLANE}) \) also includes entities being far off the original entity like \( \text{PERSON} \) and \( \text{MARC EHRIG} \). However their weight diminishes with a factor of 1/2 for each distance step. To summarize the scores of the related entities within one set three different approaches were chosen:

**Definition 9 (Entity score).** The score \( r(e) \) of a core entity is defined as follows:

- Adding up the marks of each set entity, viz. \( r(e) = \sum_{e \in R(e)} k_e \times w_e \), with \( r(e) \) being the relevance score of entity \( e \), \( R(e) \) representing one of the defined
relevance sets \( R_{e} \), \( R_{e} \), \( R_{e} \), \( R_{e} \), \( e \), the entity within the relevance set, \( w_{e} \), the weight and \( k_{e} \), the result of the entity lookup obtained by step I.

- Taking the maximum lookup result from the set of entities, viz. \( r(e) = \max_{e \in R(e)} k_{e} \times w_{e} \).
- The threshold lookup is similar to adding up all entities but only includes the entities above a defined threshold, viz. \( r(e) = \sum_{\forall e \in \{ \text{s} \in R(e) \mid (k_{e} > t) \}} k_{e} \times w_{e} \), with \( t \) being the threshold.

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. If several entity lookups were used the final relevance is calculated by adding up those as well, for example summarizing the results of metadata and free-text lookup.

**Definition 10 (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 view his documents and the crawler can optimize its search.

5 Empirical Evaluation Study

In our work we have performed an empirical evaluation study to get an idea on how the different measures proposed and the size of ontology and metadata available as background knowledge influences the quality of the overall focused crawling process. 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. In the following we will first shortly describe how the concrete relevance computation strategies have been used within our empirical evaluation. Then, we will provide some statistical information about the data sets (in form of three ontologies and their metadata). For each data set an evaluation scenario with several crawling runs was set up. A short description of the evaluation metric follows. Finally, we will explain the results we obtained within the three different scenarios.

**Relevance Computation Strategies.** 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 ontology and metadata 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 (contained in the ontology and the metadata) using the available background knowledge. All three use a sum function (as described in the last section) to compute the overall relevance-score.
Data sets. We used three different ontologies including metadata within our evaluation: First, the university ontology with 26 classes, 27 properties, 23 instances, and 224 lexical entries (synonyms and labels) referring to classes, properties and instances. Second, the airplane ontology with 14 classes, 18 properties, 19 instances, and 170 lexical entries. Third, the tourism ontology with 28 classes, 13 properties, 10 instances and 167 lexical entries.

Evaluation Scenarios. As mentioned above within our empirical evaluation study we tried to measure the quality of document and metadata discovery. The problem we have been confronted with at this point was that RDF-based metadata is currently extremely rare on the Web. Therefore, we restricted our attention within the evaluation to document discovery. Nevertheless, it is assumed that if there would be metadata descriptions available, the ontology-focused crawler would perform at least as well as without metadata descriptions. As mentioned above, we splitted the overall evaluation into three crawling scenarios. We here provide a description of the three different crawling runs we have performed within our empirical evaluation study.

1. The first scenario uses the university ontology. Goal for this crawling run is to find all pages about the Center of Intelligent Information Retrieval, an institute at the University of Massachusetts. Therefore CIIR, AMHERST, MASS (an instance of the class INSTITUTE) was selected as relevant. Direct lexical entries for this entity are “intelligent information” and “CIIR”. As starting URL “http://www.umass.edu/” was selected and only this domain was allowed for search.


3. The final 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 in the ontology are a little bit wider (“motel”, “inn”, “resort”, “oceanside”, “beachfront”). The start URLs are to follow: http://www.travel.org/, http://www.hotel.com/, http://www.guest.com/, http://www.nbc.com/, and http://www.byebye.com/.

Evaluation 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 evaluation metric used in our evaluation scenario is the well-known harvest rate [2, 1]. It represents the fraction of web pages crawled satisfying the crawling target among the retrieved pages. 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 therefore a sign of a good crawling run.
Experimental Results. The described scenarios have been processed several times to ensure the quality of the results. Representative for all the crawling runs the results of the last scenario are plotted. Breadth-first is identically to the x-axis. No relevant pages were found by this base line. 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. Using the whole ontology as background knowledge clearly outperforms the standard crawling approaches.

Fig. 5. Harvest rate vs. crawling steps in run 3

6 The CATYRPEL Document and Metadata Crawling System

CATYRPEL, the implementation of our document and metadata crawling framework, is an application within the KAON, the Karlsruhe Ontology and Semantic Web infrastructure\(^\text{10}\). CATYRPEL is embedded into the ON TOMAT-Ontology and Metadata Application Framework (see [5]), which is a comprehensive and flexible integrated development environment. The underlying data structure is provided by KAON-API\(^\text{11}\). Both, the ontology and metadata model as introduced in section 3 including the lexical layer approach are supported by KAON-API.

Figure 6 depicts a screenshot of the running CATYRPEL system. On the right side of the figure the SOEP ontology and metadata engineering plugin within the OntoMat

\(^{10}\) We invite the interested reader to have a look at this tool infrastructure and download the open-source software at http://kaon.semanticweb.org.

\(^{11}\) http://kaon.semanticweb.org/API
ontology and metadata application framework showing a domain ontology and metadata is depicted on the right side, on the left side the crawling starter interface is shown. The tight integration of the crawler with the ontology and metadata management component is important to allow for quick adaption and extension of the structures. To start a crawling process the user has to select the whole or only a subset of the ontology and metadata structure. Beside these structures the user has to select a relevance computation strategy, and the starting URLs have to be defined. On the left side of Figure 6 the initialization window is shown. The results of the crawling process are provided in several tabular interfaces. Besides the retrieved URLs with their specific relevance values and the most relevant ontology and metadata entities, the metadata contained in the documents and unknown but potentially relevant words may be shown to the user.

7 Related Work

The idea of combining focused crawling with metadata discovery 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. [2] 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. [4] published a paper in which they focus on measuring the user interest. Additionally they come up with some further ideas for a generic focused crawler. The idea of tunnelling through several links without actually rating them is a proposal on how to prevent remaining only in one domain. Some interesting aspects about when to terminate a focused crawl
are also mentioned. However, there is no detailed evaluation of their approach provided. The approach presented in [3] 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. [8] 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 argue that the learning approach is a more general framework than assuming a standard pre-defined structure while crawling.

We here give a short overview on existing metadata crawlers and their relationship to our framework. It is important to mention that in contrast to the metadata crawlers introduced below our proposed metadata crawling framework combines focused document crawling with the task of metadata crawling. Therefore, it uses linguistic knowledge combined with the background knowledge contained in the ontology to focus the document crawling process. The work that comes closest to our’s is the SHOE–Expose Crawler. Expose is a robot that searches for web pages with SHOE mark-up (see [6]), reads the knowledge from them, and loads it into the local knowledge base. Expose is initialized by given it a starting URL and setting limits on which servers and/or directories it is allowed to visit, thus, it can also be used to focus the knowledge gathering along a specific topic. SHOE–Expose Crawler includes some focussing heuristics for focussing the search for SHOE-based metadata. However, Expose does not include linguistic-based relevance measures. There are also no evaluation results available how Expose performs within concrete crawling runs.

DAML–Crawler is a relatively simple program that collects DAML statements by traversing WWW references and links. Content is currently gathered and collected by site, where a site is defined as a protocol://host:port triple. Content roots are used to identify sites. Only sites identified by content roots are currently processed. The processing of each page includes (i) the storage of the page source for subsequent use by search engines, etc., (ii) HTML parsing of the page to identify href’s and links. Such pages are added to the queue of the associated sites and finally (iii). Finally, the predecessor of CATYRPEL, viz. RDF–Crawler is a tool which downloads interconnected fragments of RDF from the Internet and builds a knowledge base from this data. At every phase of RDF crawling one maintains a list of URIs to be retrieved as well as URI filtering conditions (e.g. depth, URI syntax), which we observe as we iteratively download resources containing RDF. In contrast to our work DAML–Crawler and RDF–Crawler do not include any focusing mechanisms.

8 Conclusion

In this paper we have introduced a new approach for ontology-focused crawling of documents and RDF-based metadata. The framework includes means for adding a lex-
ical layer to RDF(S)-based ontologies and metadata, relevance computation strategies, an implementation, and an empirical evaluation of the overall framework. Our approach goes beyond existing metadata crawling, because it crawls document and metadata in parallel, it consequently uses ontology and metadata and their lexical layer as background knowledge for focusing the search in the Web. We have provided a quality-based empirical evaluation based on a standard evaluation metric harvest-rate. The empirical evaluation study has shown that relational-based relevance computation clearly outperforms typical breadth-first, keyword- or taxonomic-based approaches.

In the future we will have to evaluate our crawler in a larger application scenario including document and metadata discovery. 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\(^\text{15}\) and extended with a domain specific ontology. Thus, in the future, discovery of complex web service may be approached using our crawling approach.

References


\(^{15}\) www.purl.org/rss/1.0/schema.rdf