Noname manuscript No.
(will be inserted by the editor)

HS^3 - A Hybrid Semantic Search System

Gärtner Markus · Rauber Andreas · Seidel Ingo · Berger Helmut

Received: date / Accepted: date

Abstract In this paper we present the Hybrid Semantic Search System HS^3. Following a review of semantic search systems we identify existing common components influencing the architecture of HS^3. The system operates on an arbitrary ontology and related knowledge base providing a search service for a specific domain. The system uses the ontology and knowledge base to automatically acquire and integrate related domain-specific data from various Web resources. Annotation and indexing components of the system relate tex-

Gärtner Markus
Department of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstr. 9-11 / 188
A - 1040 Vienna, Austria
Tel.: +43 676 8200 1966
E-mail: gaertner@ifs.tuwien.ac.at

Rauber Andreas
Department of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstr. 9-11 / 188
A - 1040 Vienna, Austria
Tel.: +43 1 58801 18826
E-mail: rauber@ifs.tuwien.ac.at

Seidel Ingo
max.recall information systems OG
Pulverturmgasse 17/3
A - 1090 Vienna, Austria
Tel.: +43 720 978603
E-mail: ingo.seidel@max-recall.com

Berger Helmut
max.recall information systems OG
Pulverturmgasse 17/3
A - 1090 Vienna, Austria
Tel.: +43 720 978603
E-mail: helmut.berger@max-recall.com

tual representations of concepts and instances in documents to concepts and instances contained in the knowledge base, creating a Combined Index for information retrieval. This Combined Index is used by search and ranking services to perform a hybrid semantic search that amalgamates keyword-based and concept-based search strategies. Furthermore, we present the system's interface layer comprising the GWT-based Web UI that interactively assists the user during the search process. Finally, we present results of a usability evaluation and performance tests that have been conducted with HS^3.

Keywords Information Extraction · Information Retrieval · Ontologies · e-Tourism · Semantic Web · Semantic Search · Annotation · UI Design

1 Introduction

The main advantage of domain- specific search engines is that they operate on a very specific and information rich data corpus that is created and maintained by domain experts. Their disadvantage is that they only have a small data corpus compared to general-purpose search engines that use Web crawlers to fetch and index documents. Furthermore, the documents indexed by general-purpose search engines tend to reflect the opinion of many different people compared to a few people who maintain the data corpus of domain-specific search engines. Another difference between domain-specific search engines and general-purpose search engines is the timeliness of information. While general-purpose search constantly fetch the latest information most domain-specific search engines rely on information that has been extracted and maintained by domain experts.

Yet, when using automatically fetched data it is harder to distinguish valuable information from information that is not related to the domain or lacks significant information value. When large amounts of automatically fetched data need to be analyzed, mechanisms to automatically identify valuable information need to be applied. To identify domain information, the domain needs to be encoded in such a way that the system can match a document against the domain's encoding to check its relevancy. One possibility is to encode the domain as a list of keywords that represent the most common terms of the domain and use text or summarization-based classification to decide upon the relevancy of a document for a certain domain [32]. A common way to represent a domain is an ontology. Ontology knowledge can be used to identify relevant documents and encode relevant domain knowledge. A knowledge base (KB) stores instances of ontology concepts and their relations. The ontology and the KB can be used to support the identification of relevant data, the Information Extraction process and Information Retrieval process. To identify relevant data for a specific domain, ontology-focused Web crawlers have been developed [15, 27]. Algorithms that automatically widen the lexical coverage of textual representations of concepts, have been developed to help increase the capability of semantic applications to identify concepts related

to a certain document [5,14]. To extract relevant information in the crawled data, ontology-based Information Extraction techniques can be used [9,35].

The unique feature of concept-based search in contrast to keyword-based search is that it is not bound to occurrences of keywords in documents but rather uses abstract concepts such as people, countries or accommodations to search for relevant information. Most general-purpose search engines still adopt keyword-based search approaches that do not make use of domain knowledge encoded in an ontology or a KB. Systems that leverage ontologies and KBs to assist the search process are generally known as semantic search systems. These systems either solely rely on the KB or use a KB and a document corpus that is annotated with concepts from the ontology and instances from the KB. Even though the information contained in the document corpus can be used to support the search process, most systems solely rely on the KB to conduct searches and do not use a hybrid approach that leverages information from the KB and the document corpus to aid the search process.

In this paper we present HS^3, a hybrid search system, which makes use of semantic (concept-based) and document-related (keyword-based) information. It addresses shortages of current semantic and hybrid search systems, specifically concerning the index structure, data acquisition, query generation and result presentation.

Many hybrid search systems conduct their concept-based searches upon Triple Stores, which maintain structured data as RDF triples, and use an inverted index to conduct keyword-based searches in addition. The results are merged afterwards. Our approach uses an actual Combined Index structure based on SIREn [12], which stores structured data as triples and unstructured data as full text data and contextual data, which is related to entities encoded in the structured data. This index structure offers the advantage that information can be searched conjointly on the index-level without the need to merge it afterwards.

Another shortcoming of hybrid search systems is that document data, complementing the semantic information in the KB, needs to be acquired manually or semi-automatically. We propose a mechanism and supporting architecture to automatically fetch relevant unstructured document-based information from the World Wide Web to complement the semantic information stored in the KB. The system is designed in such a way that it can create a document corpus from scratch provided with an arbitrary ontology, corresponding KB and extraction rules.

A considerable shortage of current semantic search systems and hybrid search systems is their lack of usability. Most systems are only accessible via a rather complex structured query language such as SPARQL, RDQL or a proprietary query language, rendering them unusable for the average Internet user and even for some expert users. The interfaces are either overcrowded, use non-intuitive input methods or do not efficiently amalgamate concept-based and keyword-based search in case of hybrid search systems. To address these shortages we introduce a novel input mechanism for hybrid semantic search that combines the clean and concise input mechanisms of keyword-based

search engines with the expressiveness of the input mechanisms provided by semantic search engines. Furthermore, it is common practice for document-based search systems to use result ranking approaches. In contrast, semantic search systems hardly use ranking mechanisms to sort their results. HS^3 uses a ranking approach for the combined structured and unstructured data that is based on the user's query formulation and relevant data in the KB and document corpus.

HS^3 has been applied to two very different domains: the highly focused and specialized tourism domain and the more generic and open news domain. We used our e-Tourism ontology which is based on the Harmonise Ontology [17] for the tourism domain and the publicly available KIM Ontology and KB provided by OntoText[1] for the news domain. We applied the system to two different domains to test its flexibility and generality.

The remainder of this paper is structured as follows. In Section 2 we present related work and give an overview of semantic search systems. The shortcomings of current search systems are demonstrated by a scenario from the tourism domain in Section 3. In Section 4 we present the Hybrid Semantic Search System in detail. We present an overview of the architecture and discuss its components. Special focus is laid on the Combined Index structure and the Search & Ranking components. This is followed by a description of the User Interface that implements an interactive ontology-aware keyword-based input mechanism. In Section 5 the results of HS^3's precision and recall evaluation on two different datasets followed by the results of a usability evaluation and performance test are presented. The paper is concluded in Section 6.

2 Related Work

The following Section summarizes related work in the area of Web mining, Information Extraction (IE), Information Retrieval (IR) and Semantic Search. We start with a general overview of work in this area, followed by a detailed presentation of recent semantic search systems and their capabilities.

2.1 Web Mining, Information Extraction & Retrieval and Semantic Search

Kosala and Blockeel [26] suggest three categories for Web mining: Web content mining, Web structure mining and Web usage mining. The system presented in this paper uses techniques that belong to the Web content mining category. Besides the traditional Web mining, Semantic Web mining has emerged as part of the Semantic Web. Berendt et al. [2] describe the two main approaches in this area, namely to improve traditional Web mining by exploiting the new semantic structures in the Web; and to use Web mining for building up the Semantic Web. Li and Zhong [28] discuss two models that can be used to bridge the gap between Web mining and the effectiveness of using Web

[1] http://www.ontotext.com/

data. The pattern taxonomy model uses patterns instead of single words and techniques of sequential pattern mining to enhance the mining process. In contrast to the pattern taxonomy model where a pattern is a set of terms, the ontology mining model uses patterns that are groups of objects. Zhou et al. [36] used mining techniques to gather information from annotations and resources that a user has bookmarked to expand the user's search query. They present a query expansion framework which makes use of the data obtained from annotations and bookmarked resources.

Wimalasuriya and Dou [35] developed an approach that uses multiple ontologies to assist the Information Extraction process. Several Ontology-Based Information Extraction (OBIE) systems have been implemented so far but all of them use only a single ontology. They identified two major scenarios for using multiple ontologies in the same domain which are the specialization in sub-domains and the support of different perspectives by multiple ontologies. Cimiano and Staab [10] present a semi-automatic approach to generate metadata by using the Google API. Their approach is based on studies that show that collective knowledge of certain communities is often superior to individual knowledge. Dittenbach et al. [14] present an approach to automatically discover concepts from Web resources. The authors developed the system ConceptWorld to identify semantic concepts which relate to a particular source concept. Missikoff et al. [30] present the system OntoLearn that uses text mining techniques to automatically enrich an ontology. Berger et al. [3] developed an adaptive Information Retrieval system that is based on associative networks and demonstrated its application in the tourism domain. The system uses a knowledge representation model facilitating the definition of semantic relations between information items exemplified by terms of the tourism domain. Wang et al. [34] developed CE^2, which combines information retrieval and database technologies to realize a large scale hybrid search. Query execution times of CE^2 with ranking enabled are within one second while high precision and recall is maintained.

2.2 Semantic Search Systems

Castells et al. [7] developed a semantic search system that is capable of performing keyword-based and concept-based searches. Their main intent was to develop a semantic search system that retains precision and recall of keyword-based search when information in the KB is incomplete or even not available. A disadvantage of the system is the computational complexity of query processing, resulting in long response times. Bast et al. [1] developed a semantic search system named ESTER (Efficient Search on Text, Entities, and Relations) that leverages keyword-based search to speed up the retrieval process. The system can be used with an arbitrary ontology and corresponding KB. Nevertheless, only the usage of the YAGO [33] ontology and a KB, created from semi-structured information extracted of Wikipedia, has been documented so far. ESTER, similar to HS^3, uses a query suggestion mechanism. Kato et

System name	Features	Characteristics
Vector Space- based system of Castells et al. (2007)	* Can cope with missing information * Usage of arbitrary domain model * Extracts implicit knowledge from the KB	* Uses a strict Boolean model * Supports only RDQL * Response are up to 30 seconds
ESTER	* Response times that are fraction of a second * Interactive User Interface * Can operate with very big data sets	* Only application with data from Wikipedia demonstrated * Not clear how results are ranked
KIM	* Builds upon freely available components * Uses established standards * Comes with a pre-populated KB * Provides very good results on NER * System is available for research from the developers' Website * API for keyword-based and concept-based search	* Tightly coupled to the KIM Ontology and KB * No complex semantic queries supported out of the box * No combined search available out of the box * Ranking is based on entity popularity timelines analysis
NAGA	* Uses an advanced scoring model that makes use of confidence, informative -ness and compactness of information * Demonstration system available	* Only application with data from Wikipedia demonstrated * Demonstration system has high response times even for simple queries
AVATAR	* Uses the freely available UMIA framework for IE * System is specialized in the identification of entities in user queries to derive user's intent	* Only entity class recognition is performed but no annotation to specific entity instances * Demonstrated application of the system is bound to a corpus of email messages

Fig. 1 Overview of features and characteristics of the presented semantic search systems

al. [24] analyzed the usage of query suggestions by using three different data sets that were obtained form a commercial search engine. Their findings were that suggestions are used often if the main query is a rare query or a single term query. Furthermore users tend to use suggestions often if they are unambiguous or are generalizations or error corrections of the original query. Finally, the analysis implies that users use suggestions if they have clicked on several URLs in the first search result page.

The semantic annotation platform KIM developed by [31] also provides a semantic search functionality. However, KIM's main area of application is the extraction and annotation of entities in documents. GATE [11], a Natural Language Processing (NLP) and IE platform, is used by KIM for Information Extraction. Kasneci et al. [23] developed a semantic search system named NAGA. NAGA, similar to ESTER, uses the YAGO ontology, but is not limited to knowledge extracted from Wikipedia. NAGA's KB contains data derived from a number of semi-structured and unstructured Web sources such as Wikipedia and the Internet Movie Database (IMDB). Demartini et al. [13] use a set of algorithms that make use of the Wikipedia structure, page links and categories, to achieve a relatively high effectiveness of the Entity Ranking system. Similar to ESTER, the Entity Ranking System makes use of the

YAGO Ontology. The proposed algorithms are currently limited to Wikipedia and cannot be applied to the Web at large. Ganesan and Zhai [18] use opinion data to rank entities such as people, businesses and products based on existing opinions about those entities. Their results show that the usage of opinion expansion is effective for improving the ranking of entities according to a user's preferences. AVATAR is a semantic search engine that is based on a database approach [22]. The main intent of AVATAR is to explicitly model a user's intent encoded in a keyword query by using annotations. AVATAR uses the publicly available UIMA framework [16]. Figure 1 depicts an overview of features and characteristics of the discussed semantic search systems.

3 Why yet another Semantic Search System?

In this section we present examples of information needs that are common in the e-Tourism domain and suggest how these can be satisfied more accurately when using a hybrid semantic search system as an example of deploying such systems to support highly domain-specific search. An example of applying it in a more general news search domain is provided in Section 5. Consider the following scenario. A tourist who wants to travel to Tyrol with her son is searching for a suitable accommodation, preferably a child-friendly guesthouse, which offers an indoor swimming pool and is located in a place where she can go canyoning. Additionally, she heard from a friend that Mayrhofen is a beautiful place in Tyrol and would prefer to have the guesthouse located there. Therefore, her query to a general-purpose search engine might look like *child-friendly guesthouse in-door swimming pool mayrhofen canyoning*. The search engine will return documents that include these terms. In case there are documents that contain all terms they will be returned among the top documents for this search query. If there are no documents that contain all terms, documents that contain a subset of the query terms or are frequently referenced by documents deemed to be relevant will be among the top documents. The presented approach neglects that other documents might contain information about suitable accommodations as well, but do not explicitly contain these specific terms. Examples are documents that mention guesthouses by their actual name rather than the term *guesthouse* or documents that do not contain explicitly the term *mayrhofen* even though the guesthouses mentioned in the documents are located in Mayrhofen. Another drawback of the traditional search approach is that the semantics in the query cannot be exploited to find similar accommodations that might also be of interest for a tourist. In the presented scenario the tourist might also be interested in hotels, youth hostels, bed and breakfast accommodations and other accommodations which are located near Mayrhofen or at least in Tyrol even though she did not explicitly state it. A comprehensive description of traditional search approaches and new search approaches is given by [20].

Domain-specific search engines such as Tiscover have the advantage that they offer specific information that is maintained by domain experts. Further-

more, the User Interfaces of domain-specific search engines often assist the user in her search for information by offering drop-down lists and check boxes to express her search intent accurately. However, domain-specific search engines have the major drawback that they can only leverage information that is stored in their data storage, which is little compared to the information available via general-purpose search engines. In case of the given scenario the user would get no or only approximate results if the data storage holds no information about guesthouses in Mayrhofen that are child-friendly and offer an in-door swimming pool, even though there are accommodations that have these characteristics but mention them only on their Website.

Semantic search systems that rely entirely on their KB suffer from the same drawback as domain-specific search engines. In case the requested information is not contained in the KB, they cannot leverage any complementing data source to acquire it. Semantic search systems that use complementing data sources such as annotated document corpora for the search process are sparse and use the complementing data source only in case the information is not contained in their KB, even though the complementing data source holds additional information that can be used to perform a more specific search. Considering the given scenario, such a system will return information about appropriate accommodations if it is contained in the KB or in the complementing data source. However, in case part of the information is contained in the KB and part of it in the complementing data source, the corresponding documents will not be among the top ranked documents.

During literature review and the evaluation of semantic search systems we identified characteristics of current semantic search systems that possibly hinder the realization of the defined use case which are summarized in the following.

Even though for most semantic search systems it is stated that they can be used with an arbitrary ontology all of them have only been tested with one specific domain ontology. KIM is tightly coupled to the KIM Ontology and its KB. ESTER and NAGA are tightly coupled to the YAGO ontology and the KB that was generated from the Wikipedia corpus. To the best of our knowledge ESTER, the semantic search system proposed by [7], the system presented by [4] and the system presented by [19] are the only systems that provide a search service that can use a combination of keyword-based and concept-based search out of the box. However, techniques to exploit the information available in a document corpus in combination with the semantic information of a KB for conducting hybrid searches are still unexplored to a great extent. None of the presented semantic search systems provides an end-to-end process that includes all steps that are needed to automatically build a document corpus from Web resources, a KB and a Combined Index to conduct hybrid searches upon. The majority of semantic search systems use custom or complex input mechanisms, which are simply not applicable for the average Internet user and hinder the widespread adoption of these types of systems.

By creating HS^3 we aim to i) create a semantic search system that can use arbitrary ontologies, ii) automatically fetch and maintain related and relevant

information from Web resources, iii) utilize this information to enhance the search process by combining keyword-based and concepts based search and iv) interactively assist the user during the search process with an intuitive UI. Wherever possible, HS^3 uses components which are freely available and established standards such as Sesame [6], SIREn [12] and SwiftOWLIM [25]. The OWL dialect used by HS^3 is OWL Horst [21] which is fully supported by SwiftOWLIM, the semantic repository that is used by HS^3.

4 Hybrid Semantic Search System (HS^3)

In the following section the Hybrid Semantic Search System is presented. We briefly introduce the system's architecture, followed by a detailed discussion on how the semantic search capabilities are implemented. The architecture of the system is depicted in 2. The system consists of five main sets of components, namely the Persistence components, the Data Fetching components, the Annotation & Indexing components, the Search & Ranking services and the Interface Layer. The Persistence components are provided with data from the Ontology Enricher and the Transformation Engine. The Ontology Enricher adds textual representations of concepts to the ontology and the Transformation Engine transforms custom data structures to RDF and stores the RDF triples in the KB.

The Ontology Enricher issues a query to the WordNet API [29] for every concept in the ontology, creates a textual representation and stores it as part of the ontology. A textual representation of a concept contains one or more terms that describe a concept. For example terms such as *canyoning* and *rafting* are textual representations of the concept Canyoning. The Transformation Engine is an integral part of the system, because it is used to create the initial data set of the KB and can deal with different custom data structures. Therefore, the knowledge generated from this data is trusted and suits the same purpose as the pre-populated KB of KIM. We share the opinion of [31] that in case it is possible to engineer basic knowledge in advance with reasonable effort, it should be preferred over extracting and inferring basic knowledge with uncertain methods. The Persistence components comprise the ontology, the KB and the Document Store. The Document Store holds copies of fetched documents and the respective metadata. Annotations of documents are stored in the Document Store and can be modified or extended by Annotators or Indexers. Work queues are used by the Work Queue Managers to distribute work packages consisting of documents to the different components of the system. Any component exposes services that can be used by other components.

The Data Fetching components use services of the Persistence components. Data Fetchers and Metadata Fetchers read data from the KB and write fetched document data to the Document Store. The Metadata Fetcher is used to fetch metadata for instances of concepts that are stored in the KB. This information is subsequently used by the Data Fetcher to fetch data such as HTML documents and store them in the Document Store for further processing. The

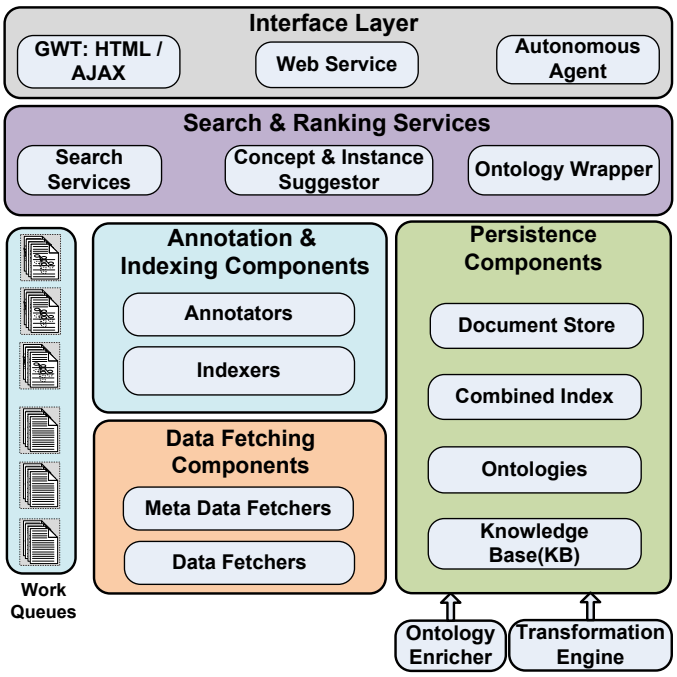

Fig. 2 System Architecture Overview

Metadata Fetcher can be equipped with plug-ins that are specialized in retrieving results for a specific concept.

The Annotation & Indexing components use data that was fetched by the Data Fetching components. Annotators use the ontologies and the KB to annotate documents in the Document Store.

Indexers operate on the ontology, KB and the Document Store which holds the annotated documents. The system includes a Semantic Indexer out of the box, but custom Indexers can be added to the system as well. The Semantic Indexer creates a Combined Index that consists of a full text index and a concept index that holds concepts, instances and their relations.

The Search & Ranking Services hold services such as the Search Services, Instance Suggestor and Ontology Wrapper, which are accessed via the Interface Layer. The Instance Suggestor service can be used to get all instances of the KB that match a certain textual representation. The Ontology Wrapper is a service to access the ontologies managed by the system. The Interface Layer offers three different types of interfaces: a GWT based Web User Interface, a Web Service Interface that enables access to the Semantic Search Service and a communication facility for autonomous software agents, which is realized as Web Service as well.

In Figure 3 an overview of the main components that are needed to create the Combined Index is presented by means of a simplified example. Consider

Component name	Input	Output
Transformation Engine	Access to relational database holding information about the entities `Elisabeth Hotel` and `Mayrhofen` OWL based tourism ontology	RDF triples describing the entities `Elisabeth Hotel` and `Mayrhofen`
Persistence Components	RDF triples describing the entities `Elisabeth Hotel` and `Mayrhofen`	RDF triples persisted in RDF Triple store.
Metadata Fetcher	Textual descriptions of the entities `Elisabeth Hotel` and `Mayrhofen`	Metadata such as the URLs, the size and the format of documents that hold information about the entities `Elisabeth Hotel` or `Mayrhofen` or both
Data Fetcher	URLs of documents that hold information about the entities `Elisabeth Hotel` or `Mayrhofen` or both that should be fetched	GATE documents created from HTML documents that hold information about the entities `Elisabeth Hotel` and `Mayrhofen`
Annotator	GATE documents created from HTML documents that hold information about the entities `Elisabeth Hotel` and `Mayrhofen` KB holding information about `Elisabeth Hotel` and `Mayrhofen`	GATE documents that hold annotations of `Elisabeth Hotel` or `Mayrhofen` or both
Indexer	Annotated GATE documents containing information about `Elisabeth Hotel` or `Mayrhofen` or both KB holding information about `Elisabeth Hotel` and `Mayrhofen`	Combined Index

Fig. 3 Overview of HS^3's components and their input and output by means of an example

a simple relational database that just contains information about the entity *Hotel* and the entity *Mayrhofen* where *Elisabeth Hotel* is located. First, the Transformation Engine reads the data from the relational database and uses the OWL-based tourism ontology to generate RDF triples that describe the entities *Elisabeth Hotel* and *Mayrhofen*. Second, the Transformation Engine uses the Persistence components to store these triples. Third, the Metadata Fetcher reads the textual representations of the entities *Elisabeth Hotel* and *Mayrhofen* from the Persistence components and fetches related metadata via the Yahoo BOSS API[2] or the Microsoft BING API[3]. This metadata holds information such as the URLs, the size and the format of documents that contain information about *Elisabeth Hotel* or *Mayrhofen* or both. Subsequently, the Metadata Fetcher stores the metadata as XML Files. Fourth, these XML files are used by the Data Fetcher to fetch the actual documents and convert them to the GATE document format. Fifth, the Annotator annotates the GATE documents with annotations referring to the instances of *Elisabeth Hotel* and *Mayrhofen* that are stored in the KB. Finally, the Indexer uses the GATE documents that hold annotations of `Elisabeth Hotel` and *Mayrhofen*, as well as the information about *Elisabeth Hotel* and *Mayrhofen* stored in the KB to create or update the Combined Index.

[2] http://developer.yahoo.com/search/boss/
[3] http://msdn.microsoft.com/en-us/library/dd251056.aspx

In the following we present the Search & Ranking Services in detail and showcase how the Google Web Toolkit (GWT) based UI is used to implement the interactive ontology-aware keyword-based input mechanism. This is followed by a comprehensive evaluation of the system and its hybrid search approach.

4.1 Search & Ranking Services

The Search Services use the Combined Index, the ontology and the KB to handle search queries. The default search service is the Semantic Search Service which supports three different types of queries. The simplest type is the keyword-only query. This query does not contain any concepts or instances that are part of the KB. In case a keyword-only query is issued the keyword index alone is used to retrieve appropriate documents. A more complex type of query is the one that contains only concepts, instances and relations among them. In case the query contains no keywords, only the index holding the indexed sub-graphs, henceforth referred to as `Index Graphs`, is used. The most complex type of query contains keywords, concepts, instances and their relations. For these queries a hybrid search on the Combined Index, consisting of the `Index Graphs`, the context data and full text data, is performed.

4.1.1 The Combined Index Structure

One purpose of the Combined Index is to relate documents to the concepts and instances which are mentioned within these documents. SIREn [12] offers the possibility to encode arbitrary tuples and store them in an index. We transform any `Index Graph` into a specific tuple structure and use SIREn to encode and store it to the Combined Index. An `Index Graph` is a star shaped graph that is transformed to its tuple representation and stored to the Combined Index. Searching these encoded graphs forms the basis for the concept-based part of the hybrid search approach. Furthermore, keywords which either describe concepts more specifically in the query are incorporated as tuples as well. For example the keyword child-friendly can be used to describe the concept `Hotel` more specifically. A keyword that describes a concept more specifically is transformed into a tuple that consists of three entries: the concept that is modified, whether it occurs in the left or right context of the concept and the keyword itself. By transforming the query into tuples that match the structure of the Combined Index, a fast retrieval of documents matching the query can be accomplished.

Multiple `Index Graphs` which have one of these concepts or instances as root node are indexed and related to the document where the root concept or instance is referenced via an annotation. An `Index Graph` is a pattern describing relevant structures and information for a specific domain in terms of concepts and their relations.

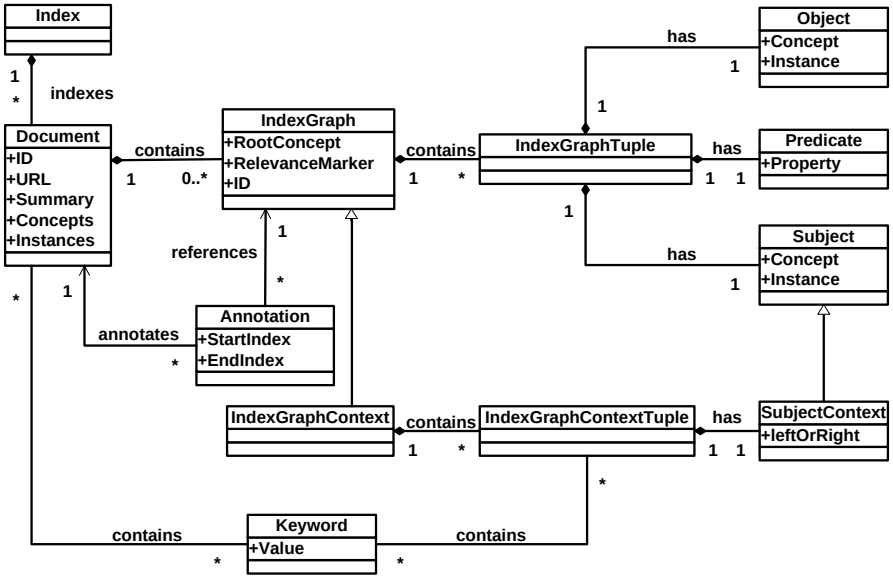

Fig. 4 The Combined Index structure

Therefore, the purpose of an `Index Graph` is to encode relevant information for a specific instance that is mentioned (annotated) within a document. It is a sub-graph of the instance's complete graph in the KB that is relevant for the retrieval process. Every concept can be equipped with an `Index Graph`. A knowledge engineer or domain expert defines the `Index Graph` for a concept, knowing what concepts and properties are relevant to users. `Index Graphs` build a central part of the Combined Index. The Combined Index structure is depicted in Figure 4. Every concept in the ontology may carry an `Index Graph` that defines the graph that should be indexed for this specific concept. This graph is used by the search mechanism to conduct searches for the specific concept. The `Index Graph` is attached as an RDF Construct Query to the concept. To store an `Index Graph` in the Combined Index it is transformed into `Index Graph Tuples`. The `Index Graph Tuples` are similar to a set of rows with columns. Every `Index Graph Tuple` has a `Subject`, an `Object` and a `Predicate`. The `Subject` contains the URI of an instance and the URI of the concept of this instance. The `Object` has the same attributes. The `Predicate` contains the `URI` of a property. An example is an `Index Graph Tuple` that has a `Subject` which contains the instance `URI` of "Hotel Elisabeth" and the concept `URI` of the concept `Hotel`, a `Predicate` which contains the property `URI` of the property `located In` and an `Object` that contains the instance `URI` of "Mayrhofen" and the concept `URI` of the concept Location. The `Index Graph Root Concept` is stored in the `Index Graph` class. The `Index`

`Graph Root Concept` is used to realize a grouping mechanism that is needed to conduct efficient approximate searches upon the index.

For example, the `Index Graph Root Concept` for the concepts `Hotel` and `GuestHouse` is `Accommodation`, which is their super-concept in the ontology as well. Therefore, if a user searches for a hotel the search algorithm would look in the field `Accommodation` instead of `Hotel`, because it is the `Index Graph Root Concept` of Hotel. Since the `Index Graphs` for guesthouses are also stored in this field, the algorithm is able to return approximate matches containing guesthouses in case no hotels exist that match the user's search criteria. Another reason for the usage of an `Index Graph Root Concept` is that an `Index Graph` needs to be defined only for a super-concept and can be inherited by all sub-concepts. Therefore, to continue the previous example, the `Index Graph` is only defined for the concept `Accommodation` and inherited to the concepts `Hotel` and `GuestHouse`. Every `Index Graph Tuple` holds information that is usually encoded in an RDF statement consisting of subject, predicate and object. But, in addition, the subject and object concept types are also included. In RDF this would be accomplished by using two additional RDF statements.

For efficiency reasons and to maintain a smaller index this information is encoded in a single tuple in the Combined Index. Any RDF graph is converted into `Index Graph Tuples` before it is indexed via SIREn. A document may be related to multiple `Index Graphs` if the root nodes of these `Index Graphs` are referenced via an annotation in the corresponding document. The root node of an `Index Graph` is the one that is not referenced by any other node within the graph The context of an instance is defined as the terms which surround the instance's annotation in a specific document. All terms to the left and to the right of an instance's annotation, within a specified window, are considered as context. These terms are indexed as `Index Graph Context Tuples`. Every `Index Graph Context Tuple` may contain multiple `Keywords` and has one `Subject Context` which defines the instance and corresponding concept that are surrounded by the contained `Keywords`. The parameter `LeftOrRight` defines whether the `Keywords` occur to the left or to the right of the subject. It can either have the value `Left Context` or `Right Context`. However, the current implementation does not differentiate between the `Left Context` and `Right Context`, but rather searches both contexts. The differentiation was designed for future use. For simplicity reasons we will refer to the `Left Context` and `Right Context` as `describedAs`.

The `Relevance Marker` of the `Index Graph` reflects the relevance of the instance, described by the `Index Graph`, in the specific document. The relevance of an instance in a specific document is calculated by multiplying its annotation's score with its annotation's position in the document (e.g. the value 10 for title or the value 1 for body) and its annotation frequency in the document. The `Relevance Marker` is used during the search process to restrict searches only to the most relevant instances of concepts within a document. This approach helps to reduce noise in the result, because common instances may be mentioned in many documents, even though the documents are mainly

about completely different instances. Documents from the tourism domain often contain names of several tourism destinations such as Salzburg, Kitzbühel or France which are annotated by the Annotation components. However, the documents that contain these terms might mainly hold information about hotels, guesthouses or apartments that are located in these tourism destinations. Consider a document that holds mainly information about the hotel Kitzhof in Kitzbühel. In that case the `Relevance Marker` for the instance Kitzhof needs to be higher than the `Relevance Marker` for the instance Kitzbühel for this specific document. However, instances of lower relevance for a specific document need to be kept in the index, because these are used for approximate matches in case no exact matches can be found. Furthermore, instances of lower relevance might just be of low relevance because the Information Extraction processes were not able to extract accurate information. Still, these instances can be useful in combination with keyword-based searches, which leverage information missed by the Information Extraction process or information which cannot be represented with the used ontology.

4.1.2 The Search & Ranking Mechanism

The Search & Ranking mechanism uses the Combined Index generated by the Indexers of the Indexing component. All queries are transformed and issued against the Combined Index. Instead of just translating the concept-based and combined queries into high-level query languages such as SPARQL or SeRQL, the query is transformed into tuples and issued against the Combined Index. During the transformation process the ontology is used to expand the query with sub-concepts and similar concepts, to return approximate results in case no exact matches have been found. The query expansion mechanism leverages so-called `Realms` to pick only meaningful concepts for the query expansion. A Realm holds concepts that are semantically related and are of interest to the user (e.g. similar concepts). To illustrate the query transformation process consider the query depicted in Figure 5. The keyword query could be something such as "child-friendly hotel providing steam bath located in mayrhofen offering rafting".

The GWT-based Web UI, described in the next section, implements the interactive ontology-aware keyword-based input mechanism that assists a user in formulating a query. In the first box of Figure 5 the query that was created by using the interactive ontology-aware keyword-based input mechanism is presented. Subsequently, this query is transformed into the graph query depicted in the second box of Figure 5. This graph representation is transformed into a tuple query, depicted in the third box of Figure 5 that consists of a disjunction of conjunctions and is issued against the Combined Index. Figure 4, depicts the transformation mechanism that is used to transform the user query graph into its tuple representation. As shown in Figure 5 the user query graph is transformed into four tuples. Whereas three tuples (nr. 1 - 3) are important for the concept-based part of the Combined Index and one tuple (namely nr. 4) is important for the keyword-based of the Combined Index. The

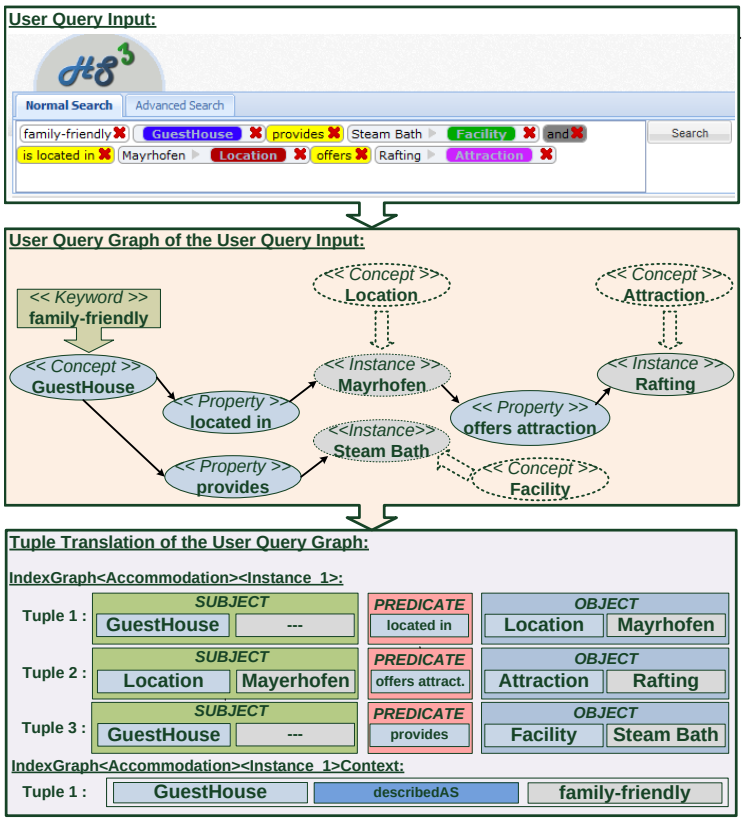

Fig. 5 Visual presentation of the query transformation process

transformation algorithm expects an ordered set of `Resources` as input. The ordered set of `Resources` is generated by the query formulation mechanism. Every `Resource` is either an instance of a concept, a concept, a property or a keyword, which can be related to other `Resources` in the query.

The transformation mechanism iterates through all Resources and checks whether the Resource is of type concept or keyword. In case the `Resource` is a concept or instance, all related `Resources` are retrieved, which can either be keywords that modify the current concept or other concepts or instances. Related concepts and instances are always related via a property. Therefore, the generated conjunction is a tuple that consists of the current `Resource's` concept or instance, the related `Resource's` concept or instance and the property that connects them. In case the related `Resource` is a keyword the tuple will contain the current `Resource's` concept or instance, a property that states that the keyword belongs to the concept's or instance's context and the keyword itself. The `Resources` and their relations are stored in a `Tuple Conjunction` object, as described in the algorithm depicted in Algorithm 1.

ALGORITHM 1: Simplified version of the query transformation mechanism

```
Input: Ordered Set OS of Resources
Output: Disjunction List DL of conjunctions
DL = new DisjunctionList();
foreach Resource r in OS do
    type = r.Type();
    if type == Concept then
        rR = r.getRelatedResources();
        tCList = createTupleConjunctions(r,rR);
        foreach TupleConjunction tC in tCList do
            DL.add(tC);
        end
    else if type == Keyword then
        DL.add(r);
    else
        skip current r;
    end
end
```

Every generated `Tuple Conjunction` is equipped with a score. The score is used by the ranking mechanism to calculate the relevance of every matched document to the user's query. The base score of a `Tuple Conjunction` in the query is determined by the types of the two Resources that are connected via the property. The highest score is assigned to a `Tuple Conjunction` that contains two instances of a concept and a property, because this `Tuple Conjunction` expressed the information need of the user in the most specific way. The next higher score is assigned to a `Tuple Conjunction` that contains an instance of a concept and a concept. The lowest score is given to a `Tuple Conjunction` that contains just two concepts and a property. Therefore, `Tuple Conjunctions` which represent the user's information need in the most specific way get assigned the highest score in the query. For example, consider a tourism ontology, where a `Tuple Conjunction` that consists of the instance "Hotel Elisabeth", the property `located in` and the instance "Mayrhofen", would express the information need of a user for a specific hotel in a specific city. According to this `Tuple Conjunction` the user would be looking for information about the hotel named "Elisabeth" which is located in Mayrhofen. Therefore, those documents which are about this specific hotel should be ranked the highest and all other hotels in Mayrhofen that might also be of interest to the user ranked lower.

However, a `Tuple Conjunction` that consists of the concept `Hotel`, the property `located in` and the instance "Mayrhofen", expresses the information need of a user in a much broader sense. Therefore, the user is not looking for a specific hotel, but rather for any hotel that is located in Mayrhofen. Hence, the user is invariant about the ranking of documents describing hotels as long as the described hotels are located in Mayrhofen. Any other hotel that is not located in Mayrhofen is of little interest to the user and is ranked lower. An example for a least specific `Tuple Conjunction` would be

one that contains the concept `Hotel`, the property `located in` and the concept `Near River`. In this case any hotel that is located near a river would be part of the result but ranked arbitrarily. In terms of query expansion, every `Tuple Conjunction` is extended with all possible concept and instance combinations. Therefore, the `Tuple Conjunction` "Hotel Elisabeth" - `located in` - "Mayrhofen" is extended with the `Tuple Conjunctions` `Hotel - located in` -"Mayrhofen", "Hotel Elisabeth" - `located in - Location` and `Hotel - located in - Location`. All `Tuple Conjunctions` get assigned a score according to their information specificity. Therefore, in the result those documents that are about the hotel named Elisabeth which is located in Mayrhofen are ranked first, followed by those about hotels in Mayrhofen, followed by other hotels that are named Elisabeth but are not located in Mayrhofen, and finally any hotel that is located in any location is listed.

Furthermore, the position of a `Tuple Conjunction` in the query graph is incorporated into its score. The farther away a `Tuple Conjunction` is from the root `Tuple Conjunction` of its query graph, the lower is its impact on the overall score of the query graph. Therefore, those `Tuple Conjunctions` which are stated first in the query graph are considered as more relevant than those at the outer border of the query graph. Consider the query graph depicted in the central box of Figure 5. In this query graph the root Concept of the graph is `GuestHouse` which is modified by the property `located in` and the instance "Mayrhofen". The instance "Mayrhofen" is modified by the property `offers attraction` and the instance "Rafting". Since "Mayrhofen" modifies the root concept of the graph and Rafting only modifies "Mayrhofen", the `Tuple Conjunction` containing the instance "Rafting" gets assigned a lower query score than the `Tuple Conjunction` containing "Mayrhofen".

However, this is an experimental feature of HS^3 which can be disabled by the user. If the user chooses to not use this feature all scores of the `Tuple Conjunctions` objects are multiplied by a factor of 1 and not by a diminishing factor that is calculated based on the concept's position in the query graph. To demonstrate the ranking functionality consider the query presented in Figure 5 and the simplified Combined Index depicted in Figure 6 which holds index information of only four documents.

As depicted in Figure 5 the user's query is transformed into the query graph and subsequently into tuples via the algorithm depicted in Algorithm 1. Subsequently, the tuples of the query are used to identify those documents that contain part or all tuples of the query in the Combined Index. In the Combined Index depicted in Figure 6 all documents hold at least one tuple that is also contained in the query. As HS^3 uses a query expansion mechanism, the query is expanded to consider not only the concept `GuestHouse` but also similar concepts such as `Apartment`, `Hotel` or `Farm` via the `Realm` functionality. For this reason *Document 1* is also considered as relevant by the search mechanism. In terms of ranking *Document 4* will be ranked first, because the tuples that are encoded in the Combined Index for *Document 4* match all query tuples and therefore the score of the document is the highest. *Document 2* will be ranked second, because it matches all but one tuple of the user query.

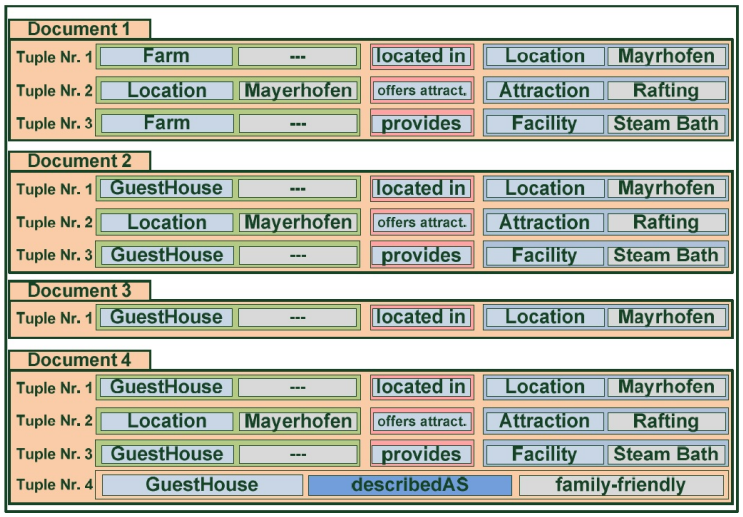

Fig. 6 A simplified Combined Index holding the index data of four documents

Document 1 will be ranked third, because it matches also all but one tuple of the user query, but gets assigned a lower score because it only contains the concept `Farm` and not `GuestHouse`. However, due to the expansion functionality, this match, which is pretty close to the user's query, is also returned. Finally, *Document 3* will be ranked fourth, because in the Combined Index only one tuple is encoded for *Document 3* that matches one of the tuples of the user's query. In case the user would have stated a specific guesthouse instance in the query, documents that hold tuples that describe this specific guesthouse would have been ranked higher than those documents that hold tuples of arbitrary guesthouses with the same criteria.

4.1.3 Conducting hybrid searches with the HS^3

Consider the scenario where a tourist wants to go for a trip to Tyrol and is looking for a child-friendly guesthouse which is located in Mayrhofen preferably. We will use the GWT-based Web UI to showcase the usage of HS^3. As one of the main issues of current semantic and hybrid search systems is their lack of usability, we developed the GWT-based Web UI, which implements an interactive ontology-aware keyword-based input mechanism. The average Internet user is used to simple input mechanisms such as the ones provided by major search engines using a simple text field and search button as input. A problem specific to hybrid search systems is that keyword-based and concept-based input needs to be supplied in separate input fields. With the GWT-based Web UI we seek to overcome this issue by providing the user with the possibility to explicitly state what terms should be treated as keywords in the query via the suggestion mechanism. The standard input mechanism

of the GWT-based Web UI looks similar to the input mechanism offered by most search engines, where a user types keywords into a text field.

Even though the input mechanism looks similar, it offers substantial help to the user by recognizing and annotating concepts and instances while the user is typing. For every recognized concept or instance in the text field the system opens a list beneath the term to let the user choose a more detailed or broader representation of a concept, hence a sub-concept or super-concept. In case the user did not intend to search for the concept or instance and was rather looking for the term in a document she can chose the `Keyword` concept from the list. Therefore, to search for a child-friendly guesthouse the user will start to type the term *child-friendly* into the text field depicted in Figure 7 (1). While the user is typing the client application checks if the lightweight ontology model, which was returned by the Ontology Wrapper service, contains any concept that is equipped with a textual representation matching the current input. Since the ontology does not contain a concept representing child-friendliness, the client application issues a call to the Instance Suggestor service to check if there is an instance in the KB with a textual representation matching the term *child-friendly*. As there is no such instance the term is automatically identified as keyword indicated by a light gray box surrounding the term *child-friendly*.

Next, the user types the term *guesth* and the suggestion mechanism suggests the concept `GuestHouse` as appropriate match. The `GuestHouse` concept has a super-concept `Accommodation`. Therefore, the client application displays a drop-down list below the current term, depicted in Figure 7 (2), that contains the concepts `Accommodation`, `GuestHouse`, similar concepts (such as `Hotel` and `YouthHostel`) and `Keyword`. The `Keyword` concept is always part of a drop-down list, because a user might be looking for the actual occurrence of this term as keyword in a document rather than for the concept itself. In this case the user chooses the `GuestHouse` concept which is shown as a blue box holding the corresponding concept. A user might choose a super-concept such as `Accommodation` instead of the actual concept when interested in other accommodation types as well.

When the user has chosen the concept the system consults the lightweight ontology model again and displays a drop-down list of properties that belong to the chosen concept depicted in Figure 7 (3). A user might either discard the suggestions by hitting the ESC key or choose a suitable property to express the information need in more detail. If the user chooses a property, it is displayed via a yellow box next to the concept. Since the user is looking for a guesthouse in Mayrhofen preferably, she will choose the `located in` property and type the term *Mayrhofen*. The client application checks the lightweight ontology again for textual representations of *Mayrhofen*. As there is no such textual representation it issues a call to the Instance Suggestor and gets in return the instances that have textual representations that equal or partly equal the term *Mayrhofen*. Now the user can choose the according instance from the list. In brackets the concept of the instance is depicted. Now the user either hits the return button or clicks the search button to issue the search request to the HS^3 server. The client application sends the query to the corresponding

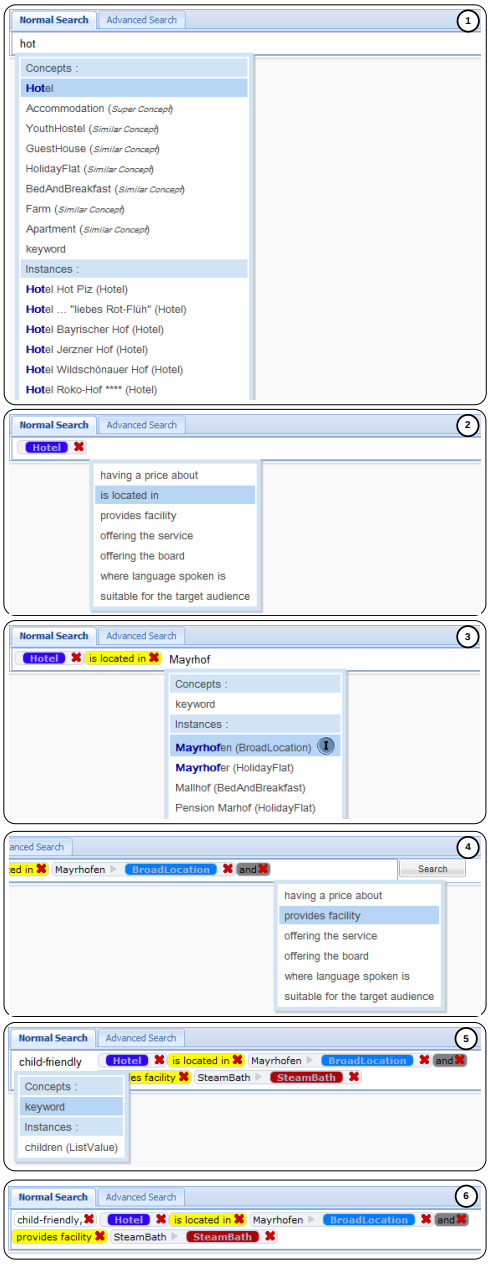

Fig. 7 Interactive GWT-based Web UI

service in the Interface Layer on the server which transforms the query accordingly so it can be interpreted by the Semantic Search Service. Now HS^3 has all information it needs to conduct a search. Since the structure includes a keyword (*child-friendly*), a concept (`Accommodation`), a relation (`located in`) and an instance of a concept (Mayrhofen) the hybrid semantic search is executed. First, the Semantic Search Service uses the index generated by SIREn to retrieve all documents that have been annotated with the `Accommodation` concept and where the instance of the mentioned accommodation is related to the instance "Mayrhofen" of the concept type `BroadLocation` via the `located in` property.

In addition the Semantic Search Service uses the keyword *child-friendly* to identify those documents that contain the keyword *child-friendly* and rank them accordingly. The result list contains links to the appropriate documents and every entry is equipped with a list of concepts and instances that are mentioned in the specific document. Interested users can click on a concept or instance which will result in a popup window displaying the RDF triples included in the KB for detailed information. Expert users may also chose the advanced input mechanism by clicking the *advanced tab* depicted in Figure 7 to issue SERQL or SPARQL queries to the KB. In return the user gets all RDF triples that match the query and in addition all documents that mention a concept or instance that is included in the result.

5 Evaluation of HS^3

We performed a 3-fold evaluation of HS^3, focusing on performance, efficiency and usability. For the performance evaluation we instructed HS^3 to automatically fetch documents to create a document corpus and Combined Index based on the KIM KB. We measured the time needed by each component of the system to perform its task. This was followed by performance tests with queries of different complexity on document corpora of different size. This gave us information about the system's scalability and the correlation between query complexity, the document corpus' size and the resulting response time. To determine the efficiency of the system we conducted a precision and recall evaluation. Finally, we conducted a usability test with 22 participants to evaluate the system's usability and user acceptance. We were mainly interested whether users of the system are able to use the novel interactive ontology-aware input mechanism that combines keyword-based and concept-based input effectively. To evaluate the applicability of the mechanism and its user acceptance we created 5 search queries, formulated in natural language, and asked the participants to formulate the search queries with the novel user interface. The data and query sets used in the evaluations are available from our project homepage[4] to serve as benchmark for further evaluations.

[4] http://www.ifs.tuwien.ac.at/ir/hybridsearch

5.1 Performance

For the purpose of evaluating the performance of the Metadata Fetcher, Data Fetcher, Annotator and Indexer components we used the KIM KB to create a corpus consisting of 167,029 unique news documents. We configured HS^3 to use one Metadata Fetcher, five Data Fetchers, one Annotator and one Indexer. We configured HS^3 to sequentially execute the metadata fetching, data fetching, annotating and indexing tasks to identify the time needed per component to perform its task. The performance tests were conducted on a commodity notebook with a 2 Gigahertz Dual Core Processor and 4 GB of RAM. It took the Metadata Fetcher about 20 minutes to fetch the metadata of approximately 200,000 news documents via the Microsoft Bing News API containing information about one or more entities stored in the KIM KB. Of the approximately 200,000 news documents that were identified by the Metadata Fetcher some were duplicates and others not available anymore, resulting in 167,029 unique documents that had been actually fetched, parsed and saved as GATE Documents to the hard disk. The five Data Fetchers needed approximately 8 hours to perform the fetching, parsing and storing of documents to the hard disk. The Annotator needed approximately 20 hours to annotate all documents and the Indexer approximately 10 hours to index all documents. In summary it took HS^3 approximately 38 hours to generate the document corpus and corresponding Combined Index when the components were running sequentially. By using more than one Annotator and Indexer the tasks can be distributed to reduce the time needed for annotating and indexing documents. Furthermore, it is possible to use computers with higher processing power or distribute the components of HS^3 to different servers to reduce the time even more.

5.2 Precision and Recall

The difficulty of evaluating precision and recall for semantic search systems and hybrid search systems arises from the fact that no public test-datasets are available that can be used for an unbiased performance comparison among competing systems. For the evaluation of precision and recall of traditional IR systems the widely accepted TREC dataset, which includes a document collection, a set of queries and judgments representing the ground truth, can be used. However, the TREC dataset is only of limited use to compare semantic systems because a related and publicly available ontology and KB as well as a corresponding ground truth would be needed. Even though precision and recall can be evaluated by using a system's primary dataset, the precision and recall results are not directly comparable to the results of other semantic search systems as long as they use different datasets. For an accurate and unbiased comparison the systems that are compared need to operate on the same document corpus, the same ontology and KB, use the same queries and need to have the same relevance judgments (ground truth) for documents. Even though there are some public ontologies available such as the KIM On-

Query Nr.	Ontology	Query
8	e-Tourism	Give me information about hotels in Salzburg that are equipped with a whirlpool
9	e-Tourism	Give me information about guesthouses in Mayrhofen that are located near a mountain
17	News (KIM)	Give me news about organizations that are located on the Cayman Islands and are traded on the NASDAQ
18	News (KIM)	Give me news about market research reports of organizations that are located on the Bermudas

tology or the YAGO Ontology, either a publicly available document corpus or a corresponding ground truth data is missing.

However, to be able to evaluate precision and recall of HS^3 we decided to make use of two different datasets. The first dataset, henceforth named the e-Tourism dataset, was created by using our e-Tourism ontology which is based on the Harmonise Ontology [17] and the KB which was created from the Tiscover database. We instructed HS^3 to automatically fetch documents that contain information about tourism accommodation and destinations from the World Wide Web. To actually compare precision, recall and the overall performance of HS^3 to other hybrid search systems we had to ensure that we use a publicly available ontology, KB and document corpus that is used by at least one other hybrid search system for the second dataset. Furthermore, we had to ensure that the systems we compare are able to cope with similar type of queries. We chose the system presented by [7], because they make use of the publicly available KIM Ontology and KB and their system can cope with queries of similar type. The authors also linked to a document corpus that consists of 145,316 news documents which they have used for the precision and recall evaluation of their approach. Unfortunately, the referenced news document corpus is not available anymore. Therefore, we instructed HS^3 to automatically build a corpus containing news documents, using the Microsoft Bing Service. As Bing only returns documents within a certain date range, HS^3 were able to fetch only about 167,029 unique documents. However, the generated corpus exceeds the news corpus used by [7] in size and should be sufficient to compare the performance of both systems.

Another reason for choosing two datasets was that the ground truth for a small dataset can be determined more accurately than the ground truth for a large dataset. To generate an accurate ground truth for a large dataset a multitude of documents would need to be evaluated manually regarding their relevance for multiple queries. This is somewhat impossible for big datasets and therefore different automatic or semi-automatic algorithms need to be used to identify relevant documents in this datasets to get an approximate ground truth. However, for reasonably small datasets the ground truth can be generated manually. Therefore, we choose a small dataset, namely the e-Tourism dataset and a big dataset, namely the news dataset, and took the average of both precision and recall evaluations to get a more accurate prediction of the overall precision and recall of HS^3.

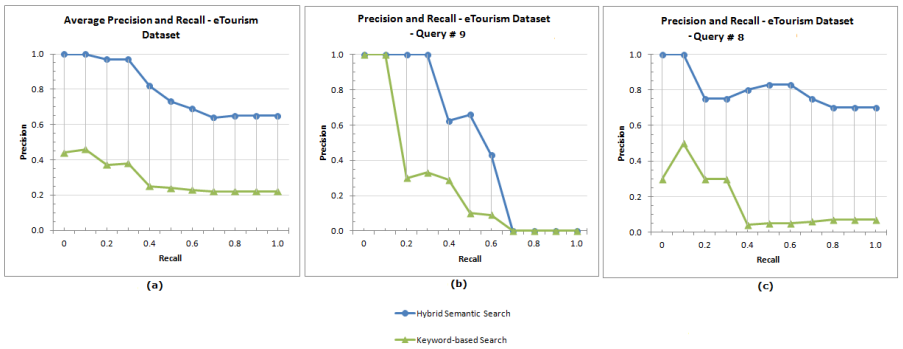

Fig. 8 Precision and Recall e-Tourism Dataset

We defined 18 queries for the e-Tourism dataset and the news dataset in total. The four queries that are discussed in more detail are listed in Table 5.2 and the rest is available on the project homepage[5]. Figure 10 (a) depicts the average precision and recall curve of the e-Tourism dataset. Furthermore, the precision and recall curves (b, c) of two specific queries out of the set of 10 queries are depicted in Figure 10 and discussed in the following.

e-Tourism Dataset : Query # 9 (Figure 10 b). The standard combined query executed on the e-Tourism dataset contains the information need "reviews about guesthouses that are located in Mayrhofen and that are nearby a mountain". As the ontology does not contain the concept mountain, the hybrid semantic search of HS^3 leverages the contextual and full text information in the Combined Index as complement to retrieve documents. The hybrid semantic search of HS^3 maintains a high precision for the given information need, because most guesthouses in Mayrhofen have been correctly annotated by the Annotator. Some have been incorrectly annotated and others where missed due to mutated vowels in their name. However, due to the combined approach the documents mentioning guesthouses which were missed by the annotator are still returned because they contain the terms "guesthouse", "mayrhofen" and "mountain" in the full text section of the Combined Index. The keyword-based search delivers poor results for higher recall values. The reason is that the keyword-based search is not able to retrieve most of the documents mentioning guesthouses, because they are mentioned with their actual name such as "Länderhof" or "Eckartauerhof". However, since the query also includes the terms "guesthouse", "mountain" and "mayrhofen" the keyword-based approach can still leverage this information to return some relevant documents.

e-Tourism Dataset : Query # 8 (Figure (Figure 10 c). The standard query executed on the e-Tourism dataset contains the information need "reviews about guesthouses that are located in Salzburg and that have a whirlpool". The hybrid semantic search provides not all relevant results be-

[5] http://www.ifs.tuwien.ac.at/ir/hybridsearch

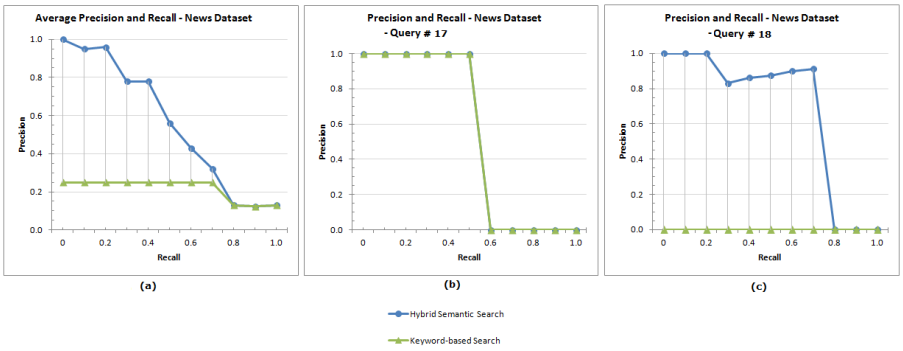

Fig. 9 Precision and Recall News Dataset

cause some guesthouses matching the criteria have not been annotated due to a mutated vowel as part of their name or encoding problems in the fetched documents. However, those documents that where correctly annotated and contain guesthouses matching the criteria are returned. Even documents that are relevant but do not hold information about the facilities of a guesthouse are returned among the top ones in case the KB states that the specific guesthouse has a whirlpool. As most documents mention guesthouses with their actual name or do not contain the term whirlpool the keyword-based approach returns only those documents that contain the terms "guesthouse", "whirlpool" and "salzburg", which explains the high precision values at low recall levels. However, in this scenario the hybrid semantic search approach cannot benefit from the full text information in the index to still return all relevant documents.

Figure 9 (a) depicts the average precision and recall curve of the news dataset. Furthermore, the precision and recall curves (b, c) of two queries out of a set of 8 queries are depicted in Figure 9 and discussed in the following.

News Dataset : Query # 17 (Figure 9 b). The standard query executed on the news dataset contains the information need "News about organizations that are traded on NASDAQ and that are located on the Cayman Islands". The corresponding precision and recall curve shows that the hybrid semantic search approach is superior for this type of information need. The keyword-based search retrieves no relevant documents. The keyword-based approach will return documents that contain the terms NASDAQ, organization and Cayman Islands. However, these documents are not relevant. The relevant documents in the corpus do not contain any of these terms. The relevant documents just contain the name of the organization such as "Garmin Ltd". However, the keyword-based approach returns documents mentioning companies that are traded on NASDAQ because both terms occur in the document, but these companies are not located on the Cayman Islands.

The hybrid semantic search approach returns relevant documents because the Combined Index holds information such as that the annotation of the

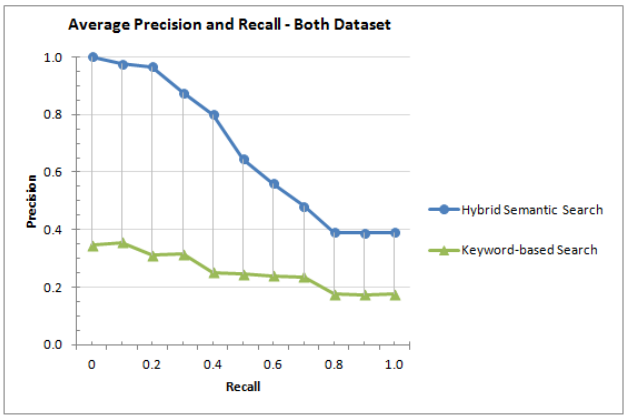

Fig. 10 Precision and Recall e-Tourism Dataset

phrase "Garmin Ltd" in a specific document refers to the public company Garmin Ltd, which is located on the Cayman Islands and traded on NASDAQ. Still, some documents are missed because the companies are not part of the KB.

News Dataset : Query # 18 (Figure 9 c). The standard combined query executed on the news dataset contains the information need "News about *market research reports* from organizations that are located in Bermuda". As the ontology does not contain the concept market research report, the hybrid search of HS^3 leverages the contextual and full text information in the Combined Index to retrieve documents that are relevant. It can be seen that the hybrid search approach of HS^3 returns relevant documents at low recall. However, since not all companies that are located in Bermuda are contained in the KB the precision drops at higher recall. The keyword-based approach, on the other hand, benefits from the fact that several relevant documents contain the phrase "market research report" and "Bermuda". In addition, documents that contain the phrase "market research report" contain mainly information about specific organizations and eliminate the need for the term organization to be part of the document. Hence, also relevant documents that mention organizations with their actual names are returned. In this specific case the keyword-based approach has the same precision and recall as the hybrid semantic search approach. However, if the information of all companies located in Bermuda would be stored in the KB, the hybrid search approach of HS^3 could still maintain a high precision at higher recall rates.

The average precision and recall curve of the e-Tourism dataset (Figure 10 a) and the news dataset (Figure 9 a) shows that the hybrid semantic search approach is in general superior to the keyword-based approach. The high precision of the hybrid search approach on the e-Tourism dataset results from the combined search approach and the sophisticated annotation and extraction rules that have been specifically tailored to the tourism domain.

Query Nr.	Query Type	Query Instructions
1	CSI	Search for an `Apartment`.
2	CST	Search for a `Hotel` located in Mayrhofen.
3	CC	Search for a `Farm` located in a `Location` where `English` is spoken.
4	CC	Search for a `Hotel` located in Retz that offers a `Telephone Service`.
5	CC & K	Search for a `Farm` where `English` is spoken and that has a `Garden`. Furthermore, the `Farm` should be described as quiet on the Webpage where it is mentioned.

Figure 10 depicts the average precision and recall curve that is obtained by combining the average precision and recall curve of the e-Tourism and news dataset. Comparing the precision and recall curve of the news dataset to the average precision and recall curve presented by [7], which was also determined by using the KIM Ontology, the corresponding KB and a news document corpus as dataset, it can be seen that HS^3's approach can compete with their approach in terms of precision and recall. However, the distinction between the two systems is that HS^3 needs at most 0.618 seconds and on average only 0.218 seconds to return the result for a query on a corpus of 167,029 annotated news documents, but the approach presented in [7] needs up to 30 seconds on a corpus of 145,316 annotated news documents. However, as discussed earlier, the precision and recall comparison should be treated with caution, because even though the systems use the same ontology and KB, the content of the document corpus used for the evaluation is different and not exactly the same queries and relevance judgments (ground truth) were used for both systems. Furthermore, the accuracy of the ground truth of the news corpus is not as accurate as the one used for the e-Tourism corpus because semi-automatic approaches have been used to create it. For an accurate comparison the same publicly available document corpus, ground truth and queries need to be used.

5.3 Usability and User Acceptance

We defined different queries in natural language and asked the test persons to formulate them using the system's interactive ontology-aware input mechanism. For this purpose we used our e-Tourism ontology and the corresponding KB. Table 5.3 lists the queries in natural language. The query type abbreviation CSI stands for a Concept Simple query, CST for a Concept Standard query, CC for a Concept Complex query, CC & K for a Concept Complex query including keywords and CSI & K for a Concept simple query including keywords. The test persons were asked to rate the difficulty of formulating a query with the interactive ontology-aware input mechanism with either "very easy", "easy", "OK", "hard" or "very hard".

The participants got a short introduction in form of a tutorial before they started with the evaluation. Of the 22 test persons who participated in the evaluation 16 were male and 6 were female. They were between 22 and 52

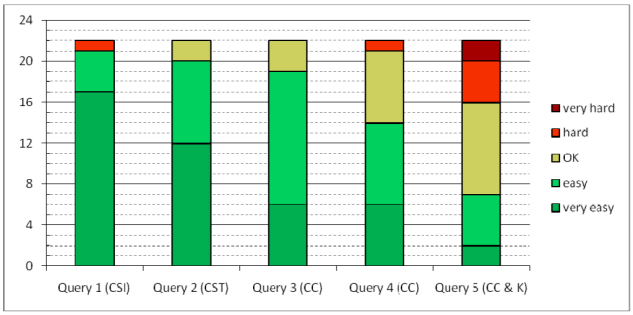

Fig. 11 Usability Evaluation Results

years old and stated that they use the Internet on a moderate or regular basis. The majority of participants were either working in software development, customer service, consulting, research, management, logistics or enrolled as students at a university. Figure 10 depicts the results of the usability evaluation. Every query is represented as one bar in the diagram and separated into distinct areas according to the difficulty ratings stated by the participants.

For example, 17 participants stated that the formulation of Query 1 was "very easy", 4 that it was "easy" and one that it was "hard". The results suggest that the majority of participants had no difficulties using the interactive ontology-aware interface. Most participants rated the difficulty of query formulation with either "very easy", "easy" or "OK".

In addition to the query formulation tasks the test persons were asked what the liked about the interactive interface, what they did not like and what they would improve. The majority of test persons stated that they liked the sleek and thin design of the interface which is not overloaded and suits the purpose of performing semantic searches well. Furthermore, they stated that it is intuitive to use and very powerful due to its suggestion mechanism and structured query formulation mechanism. A couple of test persons stated that they found the different coloring schemes very helpful, because they knew at the first glance whether they deal with a concept, an instance, a property or a keyword. All participants found the concept, instance and property suggestion mechanism either very helpful or helpful.

Asked about what they did not like about the interface a couple of test persons stated a technical problem which can be attributed to specific characteristics of the Internet Browser used.

6 Conclusion & Future Work

In this paper we have presented the Hybrid Semantic Search System and its application in two different domains. We pointed out the advantages and disadvantages of general-purpose search engines and domain-specific search

engines. Based on a scenario from the tourism domain we identified shortcomings of current search engines, which do not leverage semantic information to enhance Information Extraction and Retrieval. Furthermore, recent semantic search systems, their features and drawbacks have been presented and applied to the scenario. We identified the main components of existing semantic search systems, which also influenced the architecture of HS^3. This was followed by a presentation of the actual implementation of HS^3.

We then introduced the Combined Index and `Index Graphs`, which are used by the Semantic Search Service to conduct hybrid searches that can be either keyword-based, concept-based or a combination of both. We showcased how hybrid semantic searches are conducted by using the GWT-based Web UI of HS^3. We presented the results of performance tests, a precision and recall evaluation and a usability evaluation. The majority of test persons was able to use the novel interactive ontology-aware interface without difficulties and liked its intuitive interface as well as its suggestion mechanisms.

Regarding future work, we plan to demonstrate the application of HS^3 to additional domains and extend the suggestion mechanism of the interactive ontology-aware interface with a context-aware suggestion mechanism. Finally, we plan to give users the possibility to annotate their query with concepts and instances, which are not part of the suggestion list, to provide the system with additional information. This information can be used by the Annotation & Indexing components to annotate occurrences of previously unknown instances or identify known instances more accurate. Furthermore, we would like to apply HS^3 to a publicly available Ontology, KB and related document corpus with according relevance judgments (ground truth) to do a more accurate prediction of HS^3's precision and recall and compare it with other systems.

References

1. Bast, H., Chitea, A., Suchanek, F., Weber, I.: Ester: efficient search on text, entities, and relations. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '07, pp. 671–678. ACM, New York, NY, USA (2007). DOI 10.1145/1277741.1277856. URL http://doi.acm.org/10.1145/1277741.1277856
2. Berendt, B., Hotho, A., Stumme, G.: Towards semantic web mining. In: Proceedings of the First International Semantic Web Conference on The Semantic Web, ISWC '02, pp. 264–278. Springer-Verlag, London, UK, UK (2002). URL http://dl.acm.org/citation.cfm?id=646996.711414
3. Berger, H., Dittenbach, M., Merkl, D.: An adaptive information retrieval system based on associative networks. In: Proceedings of the first Asian-Pacific conference on Conceptual modelling - Volume 31, APCCM '04, pp. 27–36. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2004). URL http://dl.acm.org/citation.cfm?id=976297.976301
4. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid search: effectively combining keywords and semantic searches. In: Proceedings of the 5th European semantic web conference on The semantic web: research and applications, ESWC'08, pp. 554–568. Springer-Verlag, Berlin, Heidelberg (2008). URL http://dl.acm.org/citation.cfm?id=1789394.1789446
5. Bonino, D., Corno, F., Pescarmona, F.: Automatic learning of text-to-concept mappings exploiting wordnet-like lexical networks. In: Proceedings of the 2005 ACM symposium

on Applied computing, SAC '05, pp. 1639–1644. ACM, New York, NY, USA (2005). DOI 10.1145/1066677.1067050. URL http://doi.acm.org/10.1145/1066677.1067050
6. Broekstra, J., Kampman, A., Harmelen, F.v.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: Proceedings of the First International Semantic Web Conference on The Semantic Web, ISWC '02, pp. 54–68. Springer-Verlag, London, UK, UK (2002). URL http://dl.acm.org/citation.cfm?id=646996.711426
7. Castells, P., Fernandez, M., Vallet, D.: An adaptation of the vector-space model for ontology-based information retrieval. IEEE Trans. on Knowl. and Data Eng. **19**(2), 261–272 (2007). DOI 10.1109/TKDE.2007.22. URL http://dx.doi.org/10.1109/TKDE.2007.22
8. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific web resource discovery. Comput. Netw. **31**(11-16), 1623–1640 (1999). DOI 10.1016/S1389-1286(99)00052-3. URL http://dx.doi.org/10.1016/S1389-1286(99)00052-3
9. Cimiano, P., Handschuh, S., Staab, S.: Towards the self-annotating web. In: Proceedings of the 13th international conference on World Wide Web, WWW '04, pp. 462–471. ACM, New York, NY, USA (2004). DOI 10.1145/988672.988735. URL http://doi.acm.org/10.1145/988672.988735
10. Cimiano, P., Staab, S.: Learning by googling. SIGKDD Explor. Newsl. **6**(2), 24–33 (2004). DOI 10.1145/1046456.1046460. URL http://doi.acm.org/10.1145/1046456.1046460
11. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: Gate: A framework and graphical development environment for robust nlp tools and applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (2002)
12. Delbru, R.: Siren: entity retrieval system for the web of data. In: Proceedings of the Third BCS-IRSG conference on Future Directions in Information Access, FDIA'09, pp. 29–35. British Computer Society, Swinton, UK, UK (2009). URL http://dl.acm.org/citation.cfm?id=2227296.2227302
13. Demartini, G., Firan, C.S., Iofciu, T., Krestel, R., Nejdl, W.: Why finding entities in wikipedia is difficult, sometimes. Information Retrieval **13**(5), 534–567 (2010). DOI 10.1007/s10791-010-9135-7. URL http://dx.doi.org/10.1007/s10791-010-9135-7
14. Dittenbach, M., Berger, H., Merkl, D.: Automated concept discovery from web resources. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, WI '06, pp. 309–312. IEEE Computer Society, Washington, DC, USA (2006). DOI 10.1109/WI.2006.45. URL http://dx.doi.org/10.1109/WI.2006.45
15. Ehrig, M., Maedche, A.: Ontology-focused crawling of web documents. In: Proceedings of the 2003 ACM symposium on Applied computing, SAC '03, pp. 1174–1178. ACM, New York, NY, USA (2003). DOI 10.1145/952532.952761. URL http://doi.acm.org/10.1145/952532.952761
16. Ferrucci, D., Lally, A.: Uima: an architectural approach to unstructured information processing in the corporate research environment. Nat. Lang. Eng. **10**(3-4), 327–348 (2004). DOI 10.1017/S1351324904003523. URL http://dx.doi.org/10.1017/S1351324904003523
17. Fodor, O., Werthner, H.: Harmonise: A step toward an interoperable e-tourism marketplace. Int. J. Electron. Commerce **9**(2), 11–39 (2005). URL http://dl.acm.org/citation.cfm?id=1278095.1278098
18. Ganesan, K., Zhai, C.: Opinion-based entity ranking. Information Retrieval **15**(2), 116–150 (2012). DOI 10.1007/s10791-011-9174-8. URL http://dx.doi.org/10.1007/s10791-011-9174-8
19. Giunchiglia, F., Kharkevich, U., Zaihrayeu, I.: Concept search. In: Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications, ESWC 2009 Heraklion, pp. 429–444. Springer-Verlag, Berlin, Heidelberg (2009)
20. Grossmann, D.A., Frieder, O.: Information Retrieval: Algorithms and Heuristics (The Information Retrieval Series) (2004)
21. ter Horst, H.J.: Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. Web Semantics **3**(2-3), 79–115 (2005). DOI 10.1016/j.websem.2005.06.001. URL http://dx.doi.org/10.1016/j.websem.2005.06.001

22. Kandogan, E., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., Zhu, H.: Avatar semantic search: a database approach to information retrieval. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, SIGMOD '06, pp. 790–792. ACM, New York, NY, USA (2006). DOI 10.1145/1142473.1142591. URL http://doi.acm.org/10.1145/1142473.1142591
23. Kasneci, G., Suchanek, F.M., Ifrim, G., Elbassuoni, S., Ramanath, M., Weikum, G.: Naga: harvesting, searching and ranking knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD '08, pp. 1285–1288. ACM, New York, NY, USA (2008). DOI 10.1145/1376616.1376756. URL http://doi.acm.org/10.1145/1376616.1376756
24. Kato, M.P., Sakai, T., Tanaka, K.: When do people use query suggestion? a query suggestion log analysis. Information Retrieval pp. 1–22 (2013). DOI 10.1007/s10791-012-9216-x. URL http://dx.doi.org/10.1007/s10791-012-9216-x
25. Kiryakov, A., Ognyanov, D., Manov, D.: Owlim – a pragmatic semantic repository for owl. In: Proceedings of the 2005 international conference on Web Information Systems Engineering, WISE'05, pp. 182–192. Springer-Verlag, Berlin, Heidelberg (2005)
26. Kosala, R., Blockeel, H.: Web mining research: a survey. SIGKDD Explor. Newsl. **2**(1), 1–15 (2000). DOI 10.1145/360402.360406. URL http://doi.acm.org/10.1145/360402.360406
27. Kumar, M., Vig, R.: Design of core: context ontology rule enhanced focused web crawler. In: Proceedings of the International Conference on Advances in Computing, Communication and Control, ICAC3 '09, pp. 494–497. ACM, New York, NY, USA (2009). DOI 10.1145/1523103.1523201. URL http://doi.acm.org/10.1145/1523103.1523201
28. Li, Y., Zhong, N.: Ontology based web mining for information gathering. In: Proceedings of the 1st WICI international conference on Web intelligence meets brain informatics, WImBI'06, pp. 406–427. Springer-Verlag, Berlin, Heidelberg (2007). URL http://dl.acm.org/citation.cfm?id=1778453.1778481
29. Miller, G.A.: Wordnet: a lexical database for english. Commun. ACM **38**(11), 39–41 (1995). DOI 10.1145/219717.219748. URL http://doi.acm.org/10.1145/219717.219748
30. Missikoff, M., Velardi, P., Fabriani, P.: Text mining techniques to automatically enrich a domain ontology. Applied Intelligence **18**(3), 323–340 (2003). DOI 10.1023/A:1023254205945. URL http://dx.doi.org/10.1023/A:1023254205945
31. Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A.: Kim a semantic platform for information extraction and retrieval. Nat. Lang. Eng. **10**(3-4), 375–392 (2004). DOI 10.1017/S135132490400347X. URL http://dx.doi.org/10.1017/S135132490400347X
32. Shen, D., Chen, Z., Yang, Q., Zeng, H.J., Zhang, B., Lu, Y., Ma, W.Y.: Web-page classification through summarization. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '04, pp. 242–249. ACM, New York, NY, USA (2004). DOI 10.1145/1008992.1009035. URL http://doi.acm.org/10.1145/1008992.1009035
33. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A large ontology from wikipedia and wordnet. Web Semant. **6**(3), 203–217 (2008). DOI 10.1016/j.websem.2008.06.001. URL http://dx.doi.org/10.1016/j.websem.2008.06.001
34. Wang, H., Tran, T., Liu, C., Fu, L.: Lightweight integration of ir and db for scalable hybrid search with integrated ranking support. Web Semant. **9**(4), 490–503 (2011). DOI 10.1016/j.websem.2011.08.002. URL http://dx.doi.org/10.1016/j.websem.2011.08.002
35. Wimalasuriya, D.C., Dou, D.: Using multiple ontologies in information extraction. In: Proceedings of the 18th ACM conference on Information and knowledge management, CIKM '09, pp. 235–244. ACM, New York, NY, USA (2009). DOI 10.1145/1645953.1645985. URL http://doi.acm.org/10.1145/1645953.1645985
36. Zhou, D., Lawless, S., Wade, V.: Improving search via personalized query expansion using social media. Information Retrieval **15**(3-4), 218–242 (2012). DOI 10.1007/s10791-012-9191-2. URL http://dx.doi.org/10.1007/s10791-012-9191-2