Automatic Keyphrase Extraction and Ontology Mining for Content-Based Tag Recommendation

Nirmala Pudota,[†] Antonina Dattolo,[*] Andrea Baruzzo,[‡] Felice Ferrara,[§] Carlo Tasso[¶]
Artificial Intelligence Laboratory, Department of Mathematics and Computer Science, University of Udine, 33100 Udine, Italy

Collaborative tagging represents for the Web a potential way for organizing and sharing information and for heightening the capabilities of existing search engines. However, because of the lack of automatic methodologies for generating the tags and supporting the tagging activity, many resources on the Web are deficient in tag information, and recommending opportune tags is both a current open issue and an exciting challenge. This paper approaches the problem by applying a combined set of techniques and tools (that uses tags, domain ontologies, keyphrase extraction methods) thereby generating tags automatically. The proposed approach is implemented in the PIRATES (Personalized Intelligent tag Recommender and Annotator TEStbed) framework, a prototype system for personalized content retrieval, annotation, and classification. A case study application is developed using a domain ontology for software engineering. © 2010 Wiley Periodicals, Inc.

1. INTRODUCTION

Given a document, identifying an automatic methodology that generates a limited set of metadata (also called keywords, tags or keyphrases), which properly describe the given content, represents an open issue and a stimulating challenge.

This request is strongly perceived on the Web, where the amount of user-generated content and its steady growth rate exacerbate the information overload, the disorientation, the cognitive overhead, and the difficulty both to retrieve interesting documents and to classify them for future uses.

[*]Author to whom all correspondence should be addressed: e-mail: antonina.dattolo@uniud.it.
[†]e-mail: nirmala.pudota@uniud.it.
[‡]e-mail: andrea.baruzzo@uniud.it.
[§]e-mail: felice.ferrara@uniud.it.
[¶]e-mail: carlo.tasso@uniud.it.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 25, 1158–1186 (2010)

View this article online at wileyonlinelibrary.com. • DOI 10.1002/int.20448

As described in Ref. 1, traditionally this issue has been approached using systems for vocabulary control (indexing languages, thesauri, more recently using also ontologies) and classification systems (hierarchical-enumerative systems, faceted classification systems, taxonomies). These methodologies require dedicated (and human) professionals: they provide construction rules for the classification, then painstakingly read, digest, and react on the document content and finally add manually metadata values. These values match both the content of the documents themselves and the expectations and slant of the collection in which the document ends.

Although the manual process usually reaches high quality levels of classification for traditional document collections, it does not scale to the humongous size of the Web, both in terms of costs, time, and expertise of the human personnel required, and as such it cannot be proficiently put into existence for the whole Web. An alternative approach, useful for the Web, in which there is nobody in the professional role and there is too much content for a single authority to classify, is the collaborative tagging.[2] Collaborative tagging have grown widely used; numerous social tagging systems, such as Del.icio.us (delicious.com/) for Web pages, Bibsonomy (www.bibsonomy.org/) and CiteULike (www.citeulike.org/) for scientific publications, Flickr (www.flickr.com/) for images has become popular thanks to the tagging feature. Using collaborative tagging, users freely determine suitable labels for their resources without relying on any predetermined vocabulary or hierarchy; they tag the content with their own vocabulary and ultimately their mental models. Tagging is a textual annotation technique based on *metadata information*; this activity may be manual if it is generated by a human user, or automatic if it is generated by dedicated software.

Users can use tags containing a unique word (keyword) or a short phrase (keyphrase) typically containing one to three words: The tags can be employed for different scopes and tasks; in particular, we list a set of 8 intents/functions related to the tagging activity[3]:

1. *classifying a content* by means of a corpus of concepts which are familiar to the user (e.g., taxonomies, thesauri, or any bag of keywords representing meaningful categories for him/her);
2. *summarizing a resource content* by means of a short list of keywords representing the user-generated content description;
3. *expressing a polarity judgment* about a content by means of proper adjectives provided as tags (e.g., "sad," "wonderful");
4. *correlating tagged resources with people and their skills* such as the level of expertise, the reputation, or the importance of a person mentioned in the resource content (e.g., "guru," "geek," "vip," ''bill-gates");
5. *creating dichotomic classification criteria* in order to describe resources as belonging or not to a particular category (e.g., "clinical"/"not-clinical," "statistical"/"not-statistical,' "accepted"/"rejected");
6. *providing a temporal information* to a resource (e.g., dates of correlated events).
7. *identifying the content ownership* of a resource[2];
8. *refining categories*.[2] Some tags do not seem to be standalone: rather than establishing categories by themselves, they refine or qualify existing categories.

Although the differences between the eight functions are sometimes difficult to detect, to some extent all these forms of tagging express a *classification intent* targeted to establish effective schemata for organizing knowledge on the Web space and to facilitate later retrievals. But, if on the one hand these classification techniques are hard to scale and expensive, the uncontrolled/unsupervised social tagging activity deals with a set of limitations,[1] such as

- *Ambiguity*: With an uncontrolled vocabulary, many tags can be ambiguous. Indeed in tags we can find the same ambiguity that we find in natural language (e.g., homonymy, polysemy, synonymy, spelling mistakes, disambiguation).
- *Undistinguished concerns*: Social tagging systems do not enforce, or even propose, a schema for distinguishing the purpose of a metadata value. Tags might be, indifferently, proper names, subject descriptors, genres, self-reminders; tangential remarks (such as colors or years for pictures).
- *Independence of terms*: Social tagging does not provide relations to connect and relate different terms: each tag is independent of the others, and no inference is possible (the structure of a tag system is "flat").
- *Effort*: Systematically (and consistently) tagging Web resources is tedious, error prone, and rather wearying.

The rest of this paper is organized as follows: Section 2 introduces the background and related work; Section 3 summarizes the original contribution of this work; Section 4 introduces the PIRATES framework, describing its architecture (Section 4.1), and the specific modules, KpEM (Keyphrase Extraction Module), which extracts potential phrases from a document in unsupervised and domain independent way (Section 4.2) and ORE (Ontology Reasoning Engine), which automatically provides new tag recommendations for a generic textual content exploiting an ontology (Section 4.3); Section 5 provides a use case scenario, which illustrates a typical interaction between the user and the PIRATES system in the field of digital libraries. Finally, Section 6 discusses the results and the evaluation of our framework, whereas Section 7 concludes the paper with a brief overview of possible applications of our approach and future research lines.

2. BACKGROUND AND RELATED WORK

One potential solution proposed in the literature for the aforementioned problems is the development of tag recommender tools that are capable of computing tags for Web resources automatically (e.g., Refs. 4 and 5). This paper describes a content-based tag recommendation method that can be applied to any textual document; our approach mainly exploits two research areas: *keyphrase extraction* and *ontology mining*. For this reason, this section covers background of the two research areas and provides the motivation behind the exploitation of both.

A *keyphrase*[a] is a short phrase (typically it contains one to three words) that provides a key idea of a document. A *keyphrase list* is a short list of keyphrases

[a] In the remainder of this paper, in our use of this term we are also referring to *keyword*.

(typically 5–15 phrases) that reflects the content of a single document, capturing, in such a way the main topics discussed and providing a brief summary of its content.

Document keyphrases are used successfully in *information retrieval (IR)* and *natural language processing (NLP)* tasks such as document indexing,[6] clustering,[7,8] classification,[9] and summarization.[10–12] Furthermore, keyphrases are well exploited for other tasks such as thesaurus creation,[13,14] subject metadata enrichment,[15] query expansion.[16,17] Recently, keyphrase extraction method is addressed also for automatic tagging task in Ref. 18.

Keyphrase extraction methods usually work in two stages:

(i) a *candidate identification* stage identifies all possible phrases from the document;
(ii) a *selection* stage selects only few candidate phrases as keyphrases.

Existing methods for keyphrase extraction can be divided into supervised and unsupervised approaches:

A) The *supervised approach* treats the problem as a classification task. In this approach, a model is constructed by using training documents, already labeled with keyphrases assigned (by humans) to them.

This model is applied to select keyphrases from previously unseen documents. Peter Turney (developer of *Extractor*[b])[19–21] is the first one who formulated keyphrase extraction as a supervised learning problem. According to Turney, all phrases in a document are potential keyphrases, but only phrases that match with human assigned ones are considered "correct" keyphrases. Turney uses a set of parametric heuristic rules and a genetic algorithm for the extraction procedure.

Another notable keyphrase extraction system is *KEA* (Keyphrase Extraction Algorithm)[22]: it builds a classifier based on the Bayes' theorem using training documents, and it uses the classifier to extract keyphrases from new documents. In the training and extraction phases, KEA analyzes the input document depending on orthographic boundaries (such as punctuation marks, newlines, etc.) and exploits two features : $tf \times idf$ (term frequency $\times$ inverse document frequency) and first occurrence of the term.

In Ref. 23, Chen et al. presented a practical keyphrase extraction system for extracting keyphrases from Web pages. Chen et al. exploit a regression model that is trained on human-labeled documents, for the extraction of keyphrases from new documents.

Hulth[24] introduces linguistic knowledge, i.e. *part-of-speech (pos) tags*, in determining the candidate sets and, she uses 56 potential *pos-patterns* in identifying candidate phrases in the text. Her experimentation has shown that, using a *pos tag* as a feature in candidate selection, a significant improvement of the keyphrase extraction results can be achieved. Another system that relies on linguistic features is LAKE (Learning Algorithm for Keyphrase Extraction)[25]: it exploits linguistic knowledge for candidate identification and it applies a Naive Bayes classifier in the final keyphrase selection. The feature selection and learning model are the two key points for any keyphrase extraction process that is treated as a classification task. Reference 26 provides a detailed survey on shallow features plus simpler methods that can enhance the extraction performance.

All the above systems need a training data in small or large extent to construct an extraction system. However, acquiring training data with known keyphrases is not always feasible and human assignment is time consuming. Furthermore, a model, trained on a specific domain, does not always yield to good classification results in other domains.

[b] www.extractor.com/

B) The *unsupervised approach*[c] eliminates the need of training data. It selects a general set of candidate phrases from the given document, and it uses some ranking strategy to select the most important candidates as keyphrases for the document. Barker and Cornacchia[27] extract noun phrases from a document and rank them by using simple heuristics based on their length, frequency, and the frequency of their head noun. In Ref. 28, Bracewell et al. extract noun phrases from a document and then cluster the terms that share the same noun term. The clusters are ranked on the basis of term and noun phrase frequencies. Finally, top-*n* ranked clusters are selected as keyphrases for the document. In Ref. 29, Liu et al. propose an unsupervised method that extracts keyphrases by using clustering techniques and assuring that the document is semantically covered by these terms. Another unsupervised method that utilizes document cluster information to extract keyphrases from a single document is presented in Ref. 30.

Employing graph-based ranking methods for keyphrase extraction is another widely used unsupervised approach, exploited in Refs. 31–33: A document is represented as a term graph, based on term relatedness, and then a graph-based ranking model algorithm (similar to the PageRank algorithm[34]) is applied to assign scores to each term. Term relatedness is approximated in between terms that co-occur each other within a predefined window size.

Keyphrase extraction systems, developed by following unsupervised approaches, are in general domain independent since they are not constrained by any specific training documents.

As discussed above, keyphrase extraction provides potential phrases that are explicitly present in the document, without referring any vocabulary (e.g., subject headings, taxonomies); but, this does not represent a solution for the well-known problems of tagging (e.g., ambiguity); so, for instance, let us consider two documents using the term "model" in two different contexts: scientific literature and fashion. In the former, the term "model" may refer to any "mathematical model," "model in software engineering," "data model," "working model" etc., where as in the later it may refer to a "fashion model." For both documents, the keyphrase extraction system can suggest the same tag "model" without clearly referring what type of model it is. This ultimately leaves the reader in ambiguity.

On the other hand, domain-specific controlled vocabularies, classification schemata, lexicon (such as WordNet), thesauri, taxonomies, and ontologies have a huge potential to improve information organization, management and understanding with their rich subjective nature. But their usage in supporting activities like tagging represents yet an open challenge. Different methodologies and approaches used in literature have been analyzed in Ref. 1: some works simply extend ontologies in a folksonomy-like approach; other works add multiple labels to ontology nodes.

Another line of research is concerned with extracting basic semantic relations from folksonomies or adding more ontology-like features to social tagging. For example, *Folk2onto*[35] maps social tags (taken from *Del.icio.us*) to ontological categories (using a Dublin Core-based ontology) to classify and give a proper structure to the tagged resources. Another system, *ePaper*[36] uses a hierarchical news ontology,

[c] Note that unsupervised approaches might use tools like POS taggers which rely on supervised approaches. However, as such tools are usually already available for most languages, we consider an approach is unsupervised if it does not make use of any training documents that have already keyphrases assigned to them.

based on the *IPTC* (*www.iptc.org*) subject codes taxonomy, as a common language for content based filtering to classify news items and to deliver personalized newspaper services on a mobile reading device. In Ref. 37, the authors propose an ontological approach in Personalized E-Learning Scenarios; in Ref. 38 the authors present a new ontology-based model for resource inventory by integrating semantic Web technologies and agents paradigm.

In the literature, there are many examples of tag recommender systems, but the major part of them do not use ontologies: *Autotag*[39] recommends tags to Weblog posts based on the tags assigned to similar Weblog posts in a given collection; it uses information retrieval measures to find similar Weblog posts. Other systems such as Ref. 40 suggest tags for new bookmarks, using textual content associated with bookmarks to model documents and users: in this case, the authors exploit the *Bibsonomy* data set, which contains Web pages and publications. But to the best of our knowledge, automatic tagging so far has not been connected to ontology mining.

3. CONTRIBUTION OF THIS WORK

The main aim of this paper is to present our unsupervised approach dedicated to recommend significant metadata, for a given Web document: Our methodology combines tags, keyphrases extraction, and ontology mining, and assists the user when (s)he tags a Web resource. To the best of our knowledge, this is totally a new perspective for tag recommendation. Using this approach, we obtain a set of the following benefits:

- provide tags that summarize the semantic content of a Web resource (this is the purpose of keyphrases);
- provide good thematic and disambiguated tags;
- use a controlled, ontology-based vocabulary, not necessarily present in the original Web resource, and classify it as a result of the automatic tagging process;
- reduce the manual effort required to tag a Web resource.

There are some important differences between our approach for keyphrase extraction and the works discussed in the Section 2. First of all, we exploited the features in the keyphrase extraction process to rank rather than to classify keyphrases. Moreover, the keyphrase extraction approach for automatic tagging discussed in the literature (e.g., Ref. 18) typically needs training documents with keyphrases already assigned. Instead, in our work, we have introduced features that combine linguistic knowledge (such as part-of-speech tags) with statistical features (such as term frequency, phrase depth, and n-grams) in determining candidate keyphrases.

The proposed approach discussed here is implemented and tested in the PIRATES framework, a prototype system for personalized content retrieval, annotation, and classification.[3,41]

The first evaluation results show that we can effectively compute relevant tags for a variety of documents with different levels of documents' subjectivity.

4. THE PIRATES FRAMEWORK

PIRATES (*Personalized Intelligent tag Recommender and Annotator TEStbed*) is a framework for text-based content retrieval and categorization that exploits social tagging, user modeling, and information extraction techniques. The main feature of PIRATES concerns a novel approach that automates in a personalized way some typical manual tasks (e.g., content annotation and tagging). In particular, it proposes an automated method to assist a user interested in tagging a Web resource, analyzing the textual content of resources, and providing new tag recommendations on the basis of an ontology. We used the ontology to examine how the knowledge incorporated in it can help in the tasks of classification and tagging.

4.1. PIRATES Architecture

PIRATES operates on a set of input documents stored in the information base (IB) repository. To classify them, it suggests some personalized tags and other forms of textual annotations (e.g., keyphrases). The input documents are then annotated with these tags, forming the knowledge base (KB) repository.

The PIRATES architecture, shown in Figure 1, is formed by three major components:

- The *Cognitive Filtering Tools* module implements IFT (Information Filtering Tool), a system based on an algorithm[42] designed to build representations of user interests (*IFT user models*). Exploiting these models, IFT provides mechanisms of relevance feedback

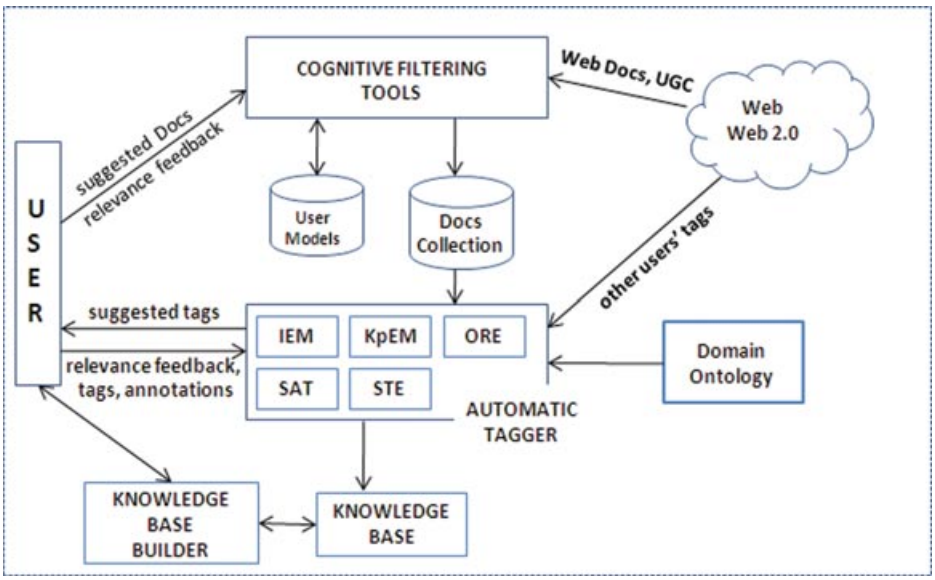

Figure 1. PIRATES architecture.

used to tune the classification of a document that belongs to an incoming stream of input documents (e.g., the results of a spidering process over the Web). The classification process produces evaluations of the relevance (in the sense of topicality) of a document according to a specific user model represented with semantic (co-occurrence) networks.
- The *Automatic Tagger* module implements a set of modules devoted to automatically annotate an incoming stream of text (the content of a document) by means of tag recommendations: IEM (Information Extraction Module) suggests named entities, KpEM (Keyphrase Extraction Module) provides keyphrases, SAT (Sentiment Analysis Tool) identifies polarity judgments, STE (Social Tagger Engine) assigns tags used by a community of Web 2.0 users, whereas ORE (Ontology Reasoning Engine) recommends tags extracted from an ontology. In this paper, we focus on the description of KpEM and ORE modules, whereas interested readers may find detailed description of the other modules in Refs. 43 and 44.
- The *Knowledge Base Builder* module organizes documents in a knowledge base repository, producing annotated documents and user conceptual maps. A more detailed description of this module is proposed in Ref. 45.

In the following sections, we present in detail the Automatic Tagger module, which is the main subject of this article. More specifically, we describe two of its components: the KpEM and the ORE modules.

4.2. Keyphrase Extraction Module (KpEM)

KpEM is a component of the Automatic Tagger that implements a family of algorithms for extracting keyphrases from textual Web documents. These algorithms can be partitioned in two different approaches:

- *supervised, domain-dependent algorithms*, which requires a supervised training procedure that exploits the explicit knowledge provided by a human (in our case, the presence of keywords pre-assigned by authors to their documents);
- *unsupervised, domain independent algorithms*, which works without both a specific domain model and any a priori knowledge about the nature of the document set.

In the first version of PIRATES system,[3] KpEM adopted only a supervised approach, incorporating a slightly modified version of KEA for the task of tag recommendation by means of keyphrases. We have now extended KpEM adding a new unsupervised algorithm for keyphrase extraction which, despite of KEA, works without requiring a training document set with preassigned keyphrases. Because of this characteristic we called this new algorithm DIKpE, Domain-Independent Keyphrase Extraction. DIKpE is illustrated in the following subsections.

4.2.1. DIKpE System

The general workflow in DIKpE is shown in Figure 2 and is illustrated in detail in the following steps. We follow three main steps:

- *Step 1*: extract candidate phrases from the document;
- *Step 2*: calculate feature values for candidates;

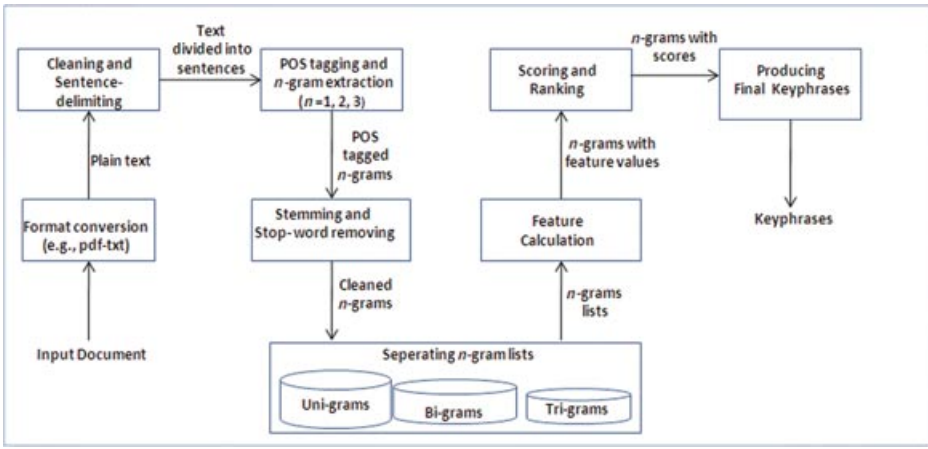

Figure 2. Workflow in DIKpE system.

- *Step 3*: compute a score for each candidate phrase from its feature values and rank the candidate phrases based on their respective scores, in such a way, highest ranked phrases being assigned as keyphrases.

Step 1: Candidate Phrase Extraction

The candidate phrase extraction step concerns several tasks such as format conversion, cleaning, and delimiting sentences, pos tagging, stemming, and properly forming n-gram lists. Each task is detailed below.

- *Format conversion*. We assume that the input document can be in any format (e.g., *pdf*), and as our approach only deals with textual input, our system first exploits document converters to extract the text from the given input document.
- *Cleaning and sentence delimiting*. The plain text form is then processed to delimit sentences, following the assumption that no keyphrase parts are located simultaneously in two sentences. Separating sentences by inserting a sentence boundary is the main aim of this step. We have used an adequate delimiter for sentence boundary. The following heuristics are applied in setting the sentence boundaries:

  – Special symbols such as '.', '@', '_', '&', '/', '-', '" are replaced with the sentence delimiter wherever they appear in the input document, but with the following exemptions:
    * The symbols '.', '@', '_', '&', '/', '-' are allowed if they are surrounded by letters or digits (e.g., *e-commerce*, *hiperlan/2*).
    * The symbol '" is allowed if it is preceded by a letter or digit (e.g., *pearson's correlation*).
  – Other punctuation marks (e.g., '?', '!') are simply replaced by sentence delimiter,
  – Apostrophes are removed and the entire input text is converted into lowercase.

  The result is a set of sentences each containing a sequence of tokens, bounded by the sentence delimiter.

- *POS tagging and n-gram extraction*. We assign a pos tag (noun, adjective, verb, etc.) to each token in the cleaned text, by using Stanford log-linear part-of-speech tagger.[d] The Stanford pos tagger uses 36 types[e] of pos tags. The assigned pos tags are later utilized in the filtration of candidate phrases and calculation of pos value feature. The next step in our procedure is to extract n-grams. We have observed that in the data set utilized for the experimentation, phrases that are constituted by more than three words are rarely assigned as keyphrases, so, in our process, we set the value of n to the maximum value three. We extract all possible subsequences of phrases up to three words (uni-grams, bi-grams, and tri-grams).
- *Stemming and stopword removing*. From the extracted n-grams, we remove all phrases[f] that start and/or end with a stopword and phrases containing the sentence delimiter. Partial stemming (i.e., unifying the plural forms and singular forms which mean essentially the same thing) is performed using the first step of Porter stemmer algorithm.[46] To reduce the size of the candidate phrase set, we have filtered out some candidate phrases by using their pos tagging information. Uni-grams that are not labeled as noun, adjective, and verb are filtered out. For bi-grams and tri-grams, only pos-patterns defined by Justeson and Katz[47] and other patterns that include adjective and verb forms are considered.
- *Separating n-gram lists*. Generally, in a document, uni-grams are more frequent than bi-grams, and bi-grams are more frequent than tri-grams and so on. In the calculation of phrase frequency (explained in the next step) feature, this shows a bias toward n-grams, which are having small value of n. To solve this problem, we have separated n-grams of different lengths ($n = 1$, $n = 2$, $n = 3$) and arranged them in three different lists. These lists are treated separately in calculation of feature sets and in final keyphrase selection. As a result of Step 1, we obtain a separate list of uni-gram, bi-gram, and tri-gram candidate phrases (with corresponding pos tags) per document after the proper stemming and stopword removal phases.

Step 2: Feature Calculation

The feature calculation step characterizes each candidate phrase by statistical and linguistic properties. Five features for each candidate phrase are computed:

- *Phrase frequency*: This feature is the classical term frequency (tf) metric, utilized in many state of the art keyphrase extraction systems,[19,24,48] in our use of this feature, instead of calculating it with respect to the whole length of the document, we compute it with respect to each n-gram list. With a separate list for each n-gram in hand, the phrase frequency for phrase P in a list L is

$$frequency(P, L) = \frac{freq(P, L)}{size(L)}$$

  where

  - $freq(P, L)$ is the number of times P occurs in L and
  - $size(L)$ is the total number of phrases included in L.

- *Pos value*: As described in Refs. 24 and 27, most author-assigned keyphrases for a document turn out to be noun phrases. For this reason, in our approach, we stress the

[d] http://nlp.stanford.edu/software/tagger.shtml.
[e] Pos tagging follows the Penn Treebank tagging scheme.
[f] In our use of this term, a phrase is n-gram with $n = 1, 2, 3$.

presence of a noun in a candidate phrase while computing a pos value for the phrase. A pos value is assigned to each phrase by calculating the number of nouns (singular or plural) normalizing it by the total number of terms in the phrase. For instance, in a tri-gram phrase, if all tokens are noun forms, then the pos value of the phrase is 1, if two tokens are noun forms, then the pos value is 0.66, and if one noun is present, the value is 0.33. All remaining phrases that do not include at least one noun form are assigned the pos value 0.25. The same strategy is followed for bi-gram and uni-gram phrases.

- *Phrase depth*: This feature reflects the belief that important phrases often appear in the initial part of the document especially in news articles, and scientific publications (e.g., *abstract, introduction*). We compute the position in the document where the phrase first appears. The phrase depth feature is computed as described in Ref. 7. The phrase depth value for phrase P in a document D is

$$depth(P, D) = 1 - \left[\frac{first\_index(P)}{size(D)}\right]$$

where

  - $first\_index(P)$ is the number of words preceding the phrase's first appearance and
  - $size(D)$ is the total number of words in D.

The result is a number between 0 and 1. The highest values represent the presence of a phrase at the very beginning of the document. For instance, if a phrase appears at 16th position, while the whole document contains 700 words, the $phrase\_depth$ value is 0.97, indicating the first appearance at the beginning of the document.

- *Phrase last occurrence*: We give also importance to phrases that appear at the end of the document, since keyphrases may also appear in the last parts of a document, as in the case of scientific articles (i.e., in the conclusion and discussion parts). Last occurrence feature is proposed in Ref. 49 for extracting keywords (i.e., n-grams where $n = 1$); however, we exploit the feature for extracting keyphrases (i.e., n-grams where $n \geq 1$). The last occurrence value of a phrase is calculated as the number of words preceding the last occurrence of the phrase normalized with the total number of words in the document. The last occurrence value for phrase P in a document D is

$$last\_occurrence(P, D) = \frac{last\_index(P)}{size(D)}$$

where

  - $last\_index(P)$ is the number of words preceding the phrase's last appearance and
  - $size(D)$ is the total number of words in D.

For instance, if a phrase appears for the last time at 500th position in a document that contains 700 words, then the $phrase\_last\_occurrence$ value is 0.71.

- *Phrase lifespan*: The span value of a phrase depends on the portion of the text that is covered by the phrase. The covered portion of the text is the distance between the first occurrence position and last occurrence position of the phrase in the document. The span feature is utilized in Ref. 49 on lexical chains[g] for extracting keywords. The lifespan

[g] A lexical chain is a set of words that are related with each other.

value is computed by calculating the difference between the *phrase last occurrence* and the *phrase first occurrence*.

The lifespan value for phrase P in a document D is

$$lifespan(P, D) = \frac{[last\_index(P) - first\_index(P)]}{size(D)}$$

where

- $last\_index(P)$ is the number of words preceding the phrase's last appearance,
- $first\_index(P)$ is the number of words preceding the phrase's first appearance, and.
- $size(D)$ is the total number of words in D.

The result is a number between 0 and 1. The highest values mean that the phrase is introduced at the beginning of the document and carried until the end of the document. Phrases that appear only once through out the document have the lifespan value 0.

As a result of step 2, we get a feature vector for each candidate phrase in the three n-gram lists.

Step 3: Scoring and Ranking

In this step, a score is assigned to each candidate phrase, which is later exploited for the selection of the most appropriate phrases as representatives of the document. The score of each candidate phrase is calculated as a linear combination of the five features. We call the resulting score value as *keyphraseness* of the candidate phrase.

The keyphraseness of a phrase P with non empty feature set $\{f_1, f_2, \ldots, f_5\}$, with nonnegative weights $\{w_1, w_2, \ldots, w_5\}$ is

$$keyphraseness(P) = \frac{\sum_{i=1}^{5} w_i f_i}{\sum_{i=1}^{5} w_i}$$

In the first stage of our research, we assigned equal weights to all features, yielding to the computation of the average. Therefore:

$$keyphraseness(P) = \frac{1}{n}\sum_{i=1}^{n} f_i,$$

where:

- n is the total number of features (i.e., five in our case),
- f_1 is the phrase frequency,
- f_2 is the phrase pos value,
- f_3 is the phrase depth,

- f_4 is the phrase last occurrence, and
- f_5 is the phrase lifespan.

A feature could have more impact than others on keyphraseness and influences how candidate phrases will be selected, as the features have not the same nature (e.g., frequency vs. pos value, depth vs. lifespan). To compensate this phenomena different weights are assigned to each feature.

For weight calculation, we are proposing a novel approach that computes associate weights to features by examining the already existing ground truth author-assigned keyphrases. For this, we utilized a publicly available keyphrase extraction data set[h50] which contains 215 full length scientific documents from different computer science subjects. Each document in the data set contains a first set of keyphrases assigned by the paper's authors and a second set of keyphrases assigned by volunteers, familiar with computer science papers. We considered author-assigned and volunteer-assigned keyphrases as ground truth keyphrases for the documents. For these keyphrases, the five feature values are computed as explained in Section 4.2. Since DIKpE is capable of extracting only phrases that are explicitly stated in the documents, from 215 documents, overall, 1000 keyphrases with corresponding feature values are computed. Highest values (i.e., near to 1) represent the goodness of the feature. The 1000 keyphrases with five feature values are represented in matrix form as follows:

$$F = \begin{bmatrix} f_{(1,1)} & f_{(1,2)} & f_{(1,3)} & f_{(1,4)} & f_{(1,5)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{(j,i)} & f_{(j,i)} & f_{(j,i)} & f_{(j,i)} & f_{(j,i)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{(1000,1)} & f_{(1000,2)} & f_{(1000,3)} & f_{(1000,4)} & f_{(1000,5)} \end{bmatrix}$$

where $f_{(j,i)}$ represents the value of the ith feature calculated for the jth keyphrase ($i = \{1, \ldots, 5\}$ and $j = \{1, \ldots, 1000\}$).

For each feature f_i the mean μ_{f_i} and the variance $\sigma^2_{f_i}$ of the vector $[f_{(1,i)}, \ldots, f_{(j,i)}, \ldots, f_{(1000,i)}]$ are calculated as follows:

$$\mu_{f_i} = \frac{\sum_{j=1}^{1000} f_{(j,i)}}{1000}$$

$$\sigma^2_{f_i} = \frac{\sum_{j=1}^{1000} f^2_{(j,i)}}{1000} - \mu^2_{f_i}$$

The mean reflects the central tendency of the feature, and the variance reflects the variability of the feature values with respect to the mean. Obviously, the feature

[h] http://wing.comp.nus.edu.sg/downloads/keyphraseCorpus/

Table I. Final Weights Assigned to the Features.

Feature name	Weight
Phrase frequency	0.10
Pos value	0.30
Phrase depth	0.32
Phrase last occurrence	0.16
Phrase lifespan	0.12

having high-mean and low-variance contributes maximum to the final keyphrase result. The weight for f_i is computed by simply dividing the mean with the variance.

In the equation form, the weight of f_i is

$$weight(f_i) = \frac{\mu_{f_i}}{\sigma^2_{f_i}}$$

The weights are normalized and assigned to the features. Our weight calculation approach resulted the highest weight to the phrase depth feature, then second highest is given to pos value, and so on for last occurrence, lifespan, frequency features, respectively. The final weights are shown in Table-I.

Experiments showed that results are better when considering the associate weights, and they are detailed in Section 6.

Producing Final Keyphrases. The scoring process produces three separate lists L_1, L_2, and L_3 containing, respectively, all the uni-grams, bi-grams, and tri-grams with their keyphraseness values. We then select some keyphrases, which are considered to be the most important from each list. In Ref. 51, Kumar and Kannan proposed a strategy for selecting final keyphrases from n-grams, after an extensive statistical analysis regarding the length of the author assigned keyphrases for scientific documents. In this paper, the documents we are utilizing for the experiment are scientific in nature, to produce the k final keyphrases, we have followed the same strategy that is proposed and utilized in Ref. 51. In every list, the candidate phrases are ranked in descending order based on the keyphraseness values. Top 20% (i.e., 20% of k) keyphrases are selected from "L_3," Top 40% (i.e., 40% of k) are selected from "L_2," and remaining 40% of rest of k keyphrases are selected from "L_1." In this way top k keyphrases for the given document are extracted.

4.3. Ontology Reasoning Engine

ORE is a software module, which extracts tags from an ontology. More specifically, it does inference over a local ontology written in OWL format, using a reasoning mechanism based on is-a relationship between the ontology concepts. In this way, it can suggest terms that are not present in the examined text document, providing also a higher level of abstraction for the recommended tags due to the inheritance hierarchy relations expressed in the ontology. The inference mechanism

is based on a matching algorithm that searches the correspondences between the tags provided by PIRATES annotators and the concepts of a domain ontology. It is possible to use any general ontology, but using the more specialized vocabulary of a domain ontology increases the likelihood to find correspondences between the terms present in both the ontology and the documents.

4.3.1. Ontology Mining to Infer New Knowledge

Our ontology-based tag recommender system works in combination with KpEM. In fact, its use is applied in cascade of the keyphrase extraction procedure executed by KpEM. Initially, for each keyphrase provided by KpEM for a given document, ORE is programmed to find the corresponding match with the terms in the ontology. ORE is useful if there exists at least one match. In such case, ORE follows a special navigation strategy to find ancestor nodes and common ancestor nodes of the corresponding matches. We have followed the spreading activation algorithm[52] to implement the navigation strategy composed by the following steps:

- For each keyphrase extracted by KpEM for a given document, the algorithm looks for a corresponding match in the ontology, retrieving its immediate superclass by following parent–child relationship.
- As second step, the retrieved superclass is marked as ontology concept mapping node.
- Then, if there are at least two ontology concept mapping nodes, it retrieves the common ancestor node for them and possibly all the nodes in the path between the ontology concept mapping nodes and the common ancestor node.

5. A USE CASE SCENARIO IN THE FIELD OF DIGITAL LIBRARIES

In the past years, we worked on the E-Dvara platform[i], an international research project in the area of digital libraries. To experiment with the PIRATES framework, we recently started to integrate its semantic services in E-Dvara,[53] approaching the vision of *semantic digital library*.[54] We found in the literature several projects that integrate ideas coming from the Semantic Web for representing, structuring, and classifying information.[55–57] Other approaches emphasize more the integration of social practices, like social and collaborative tagging, arising from the Web 2.0 experience.[58,59] Our contribution is aimed at combining these approaches, introducing in E-Dvara dedicated semantic tools for

- automatically suggesting new content related to a specific topic of interest for the user,
- recommending similar documents with respect to a search query performed by the user, and
- helping the user to better specify his information need by means of a query reformulation service.

In the rest of this section, we discuss the application of the PIRATES framework to provide the above-mentioned services in the context of digital libraries. We also

[i] http://edvara.uniud.it/india

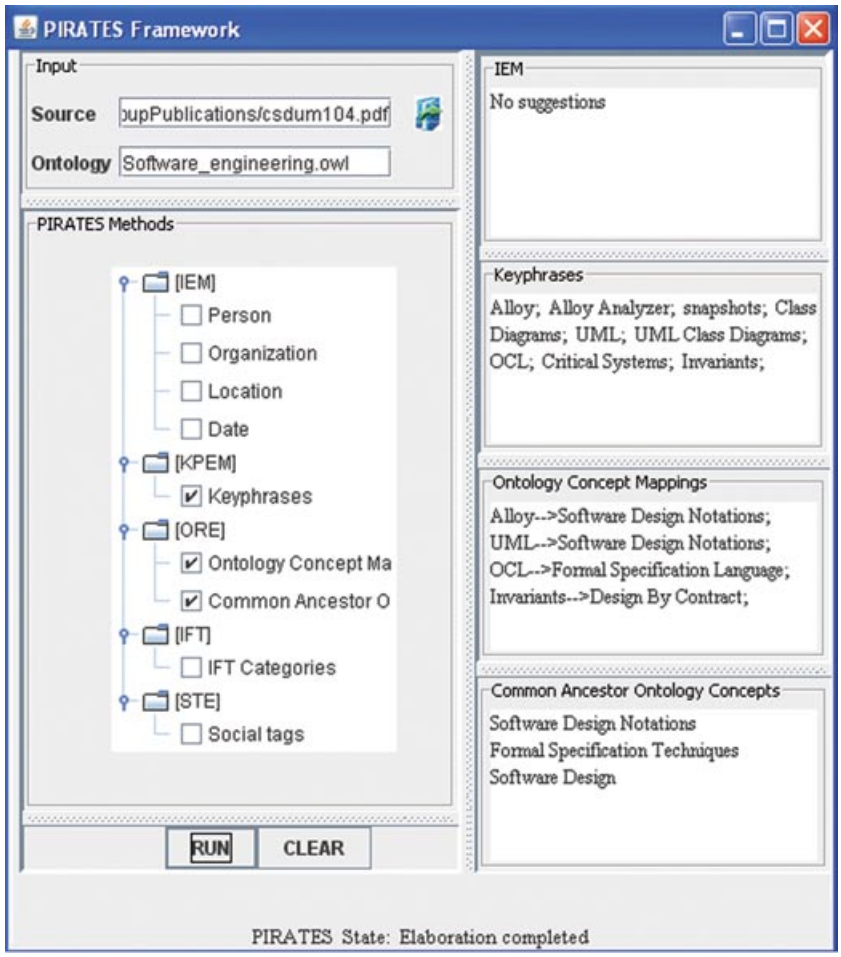

Figure 3. A screenshot of our PIRATES prototype.

highlight the role of the PIRATES annotators presented in this article: KpEM and ORE.

Suppose we have a user collecting an archive of Web documents in the field of software engineering automatically retrieved by the Cognitive Filtering Tools service embedded in the digital library. Suppose also that, one day, this tool notifies (among the others) the paper "A UML Class Diagram Analyzer".[j] To classify this new content, the user exploits two PIRATES annotators, KpEM and ORE (Figure 3). In particular, in this example, the user configured the ORE annotator to use an ontology

[j] http://twiki.cin.ufpe.br/twiki/pub/SPG/GroupPublications/csduml04.pdf.

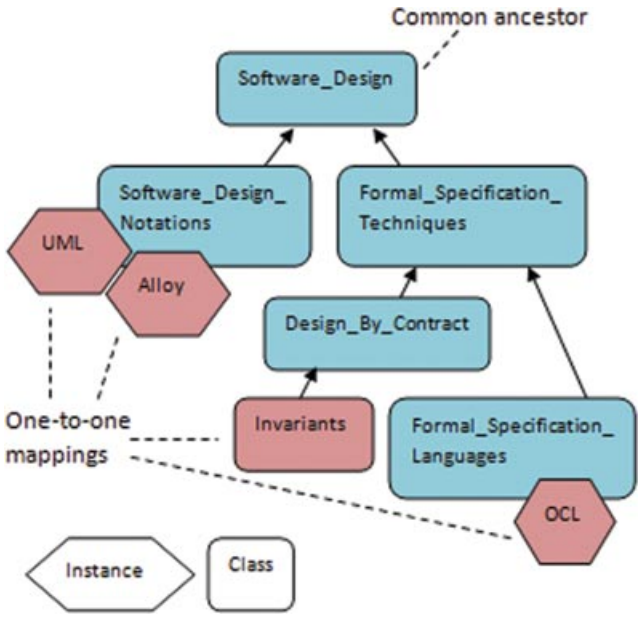

Figure 4. Ontology reasoning.

in the field of software engineering. For our tests, we extend the publicly available *Software_engineering.owl*[k] ontology including several concepts at the instance level. Using this ontology and starting from the keyphrases extracted by KpEM, ORE implements the navigation strategy described in Section 4.3. The Protégé-OWL API[l] is utilized for the implementation of the navigation strategy. The API provides classes and methods to load, save, query, and perform reasoning on OWL ontologies.

Four out of the suggested keyphrases (i.e., *Alloy*, *UML*, *OCL*, and *Invariants*) are matched by ORE with a corresponding concept (either at the class or at the instance level) in the ontology, as shown in Figure 4. Starting from these nodes, ORE uses the spreading activation algorithm to find common ancestors representing more abstract subjects. Then both one-to-one ontology mappings and common ancestors are provided to the user by PIRATES as potential tag recommendations. In this way, for the input document, ORE recommends five new tags, which are not presented in the text (i.e., Software Design Notations, Formal Specification Languages, Design by Contract, Formal Specification Techniques, and Software Design). These tags represent *abstractions* of the keyphrases extracted by the other annotators available in PIRATES. Automatic tag recommendation may lead to several improvements in

[k] The Software_engineering.owl ontology has been developed at the Curtin University of Technology and is available from the SEONTOLOGY Web site http://www.seontology.org/.

[l] Protégé-OWL API is an open-source Java library for the Web ontology language and RDF(S). It is available for download: http://protege.stanford.edu/plugins/owl/api/.

content access in several applicative contexts. In digital libraries, for example, one of such improvements is the annotation-based content recommendation of *similar documents*. For example, if a user is accessing a specific document already available in the archives of the digital library and yet tagged by PIRATES, ORE can be exploited to automatically find a set of most similar contents to present the user. Similarity, this case can be defined as the set of common tags shared by both a source content and the other contents available in the archives. In this scenario, each user can provide a query to the digital library search engine, browse the results, and in real-time, receive a list of suggested contents, which do not necessarily contain the same keywords used in the query (tags identified by PIRATES may represent concepts that are not directly referred in the resource, but obtained as a result of the ontology base inference).

Another scenario concerns the task of *query reformulation*. Semantic representation of available contents, provided manually or in an automatic way, may also be exploited to improve the effectiveness of traditional keyword-based retrieval. Indeed, semantic knowledge can be used to augment the effectiveness of traditional keyword search mechanisms, moving further toward the concept of *semantic search*. To achieve such goals, a query reformulation engine will be included in the digital library as an explicit semantic layer. Using both the contents metadata and the ontologies constituting the Knowledge Base of the platform, the query reformulation engine will intercept the requests submitted by users and suggests, in addition to the retrieved contents, a list of potentially related queries.

According to the workflow and the nature of the set of modules constituting the PIRATES framework, the query reformulation task will be based on two different kinds of knowledge: ontology-based reformulation and annotations-based reformulation. Ontology-based reformulation will be used to identify concepts similar to the terms used in the query by browsing a domain-dependent ontology used by the ORE module to annotate resources. Such concepts may be included in the query or can be used to substitute existing terms. Annotations-based reformulation, on the other hand, is based on the tags assigned to the contents retrieved using the original query; by ranking tags and looking at the most relevant ones (relevance will be defined as a tag frequency function), reformulation engine can generate a new query. Annotations-based reformulation exploits all the different kinds of annotations provided by the PIRATES framework; such approach may be seen as complementary to the one used by the ontology-based reformulation, where only knowledge occurring in the domain ontology is considered.

6. RESULTS AND EVALUATION

This section presents the results of the experimentation conducted on the tag recommendation modules and the evaluation process.

Our goal is to estimate the effectiveness of the PIRATES tag recommendation approach comparing the number of our suggested keyphrases with that of keyphrases assigned by the document's author. Simply, let $\varphi_i(D)$ be the set of i suggested keyphrases for a document D by the recommender φ, and $keyphrases(D)$ the set

of keyphrases associated with the document D by his (her) author; then, if m is the number of documents, we calculate the average number of correct keyphrases (suggested by the recommender φ when it returns i keyphrases) as follows:

$$\frac{1}{m}\sum_{D}|\varphi_i(D)\cap keyPhrases(D)|$$

The applied metric is quantitative and does not take into account the subjectivity of any human judgment about the appropriateness of the semantic relation of an automatically generated (inferred) tag with respect to a specific document. Thus, a tag suggested by PIRATES that does not match with a preassigned keyword is considered as a "bad tag" ("wrong result") in this evaluation. Clearly, this is not necessarily be the case, since in general there may exist many tags, different from the keyphrases assigned by the authors, that are equally good for a given document. By the way, the results discussed below should be interpreted as a lower-bound performance indicator (not all the "wrong recommendations" can always be interpreted as such by human users of the system). Three different experiments are conducted to test the system's performance. Publicly available keyphrase extraction data set[h] is utilized for the first two experiments. Our own data set is utilized for the third experiment for the use of ontology. Following subsections provide the complete details of the experiments.

6.1. Experiment 1

For the first experiment, we have considered keyphrase extraction works presented by Nguyen and Kan[50] and KEA[22] as baseline systems. From the available 215 documents in the data set, Nguyen and Kan has taken 120 documents to compare these with KEA. The maximum number of keyphrases for each document (i.e., k) is set to 10 in Nguyen and Kan. We have taken their results[50] as reference, and in the first experiment we have worked on 120 documents randomlly selected from the 215 documents. In all the experiments, we removed the bibliography section from each document in the data set to better utilize the *phrase last occurrence* feature. The same partial stemming strategy exploited in candidate phrase selection (see Section 4.2.1) is used also in matching correct keyphrases.

Table II shows the average number of correct keyphrases of three algorithms when 10 keyphrases are extracted from each document: The first row shows the average number of correct keyphrases, i.e., 3.03 suggested by KEA, the second row shows the average number of correct keyphrases, i.e., 3.25, suggested by Nguyen and Kan, the third rows shows the average number of correct keyphrases, i.e., 4.75, suggested by our system DIKpE when equal weights are assigned to the features, whereas the last row shows the average number of correct keyphrases, i.e., 5.04, suggested by DIKpE after assigning associate weights to the features. In either of the cases, our system significantly outperforms the other two.

Table II. Overall Performances.

System	Average number of Correct Keyphrases
KEA	3.03
Nguyen&Kan	3.25
DIKpE (before assigning weights)	4.75
DIKpE (after assigning weights)	5.04

6.2. Experiment 2

For the second experiment, we have extracted keyphrases for all 215 documents and compared our approach exclusively with the results provided by KEA as KEA is publicly available and considered as an emerging benchmark for evaluation in the literature. We have utilized a total of 70 documents (with keyphrases assigned by authors) extracted from the 215 documents data set to train the KEA algorithm. For each document, we extracted 7, 15, and 20 top keyphrases using both our approach and KEA.

The results are shown in Table III and graphically represented in Figure 5. The lowest line describes the performances of KEA which, on average, correctly recommends respectively 2.05, 2.95, and 3.08 keyphrases when it returns 7, 15, and 20 keyphrases. Performances of both DIKpE before assigning weights (the central line) and DIKpE after assigning weights (the upper line) outperform significantly the results of KEA, where DIKpE after assigning weights gives the best results.

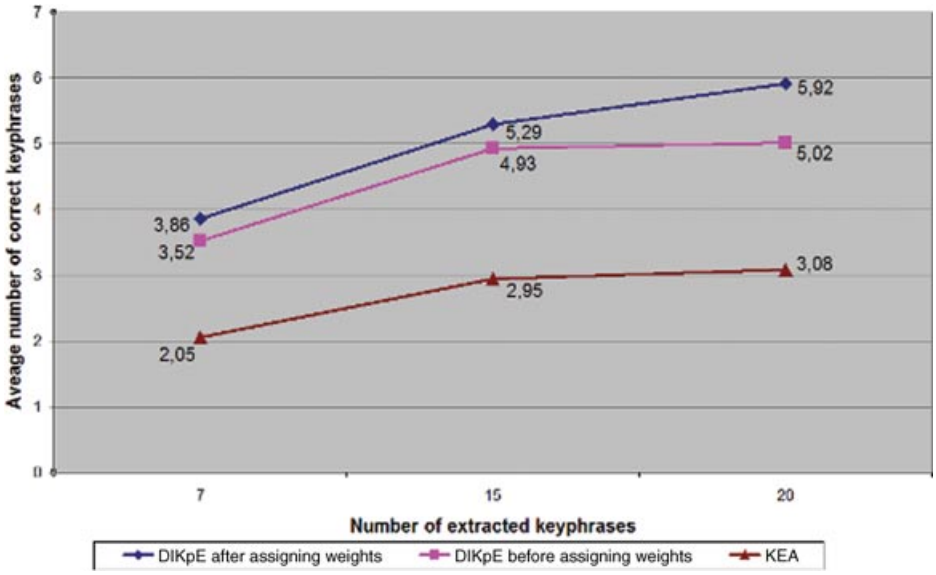

Figure 5. Performance of DIKpE compared to KEA.

Table III. Performance of DIKpE Compared to KEA.

	Average number of correct keyphrases		
Extracted keyphrases	KEA	DIKpE (before assigning weights)	DIKpE (after assigning weights)
7	2.05	3.52	3.86
15	2.95	4.93	5.29
20	3.08	5.02	5.92

6.3. Experiment 3

To utilize the SEOntology of software engineering domain[k] that is publicly available, for the third experiment we have generated our evaluation data set. To obtain the data set, we used the Cognitive Filtering Tools module and the IFT algorithm (see Section 4.1): they automatically generated a collection of scientific papers concerning various themes in the software engineering field. We selected among them, 60 documents ($m = 60$) which explicitly contained a set of preassigned keywords by authors. On the whole, this restricted data set contains a total of 289 keyphrases. The following sections detail the third experiment that utilized the ontology.

We relate the performance of our framework by comparing it with that of other recommender systems, such as KEA: we executed several experiments to evaluate the average number of correct suggestions provided by different recommenders when they suggest 3, 5, 7, and 10 keyphrases, respectively.

6.3.1. KpEM Tag Recommendations

Table IV shows the keyphrases recommended by KpEM for three sample documents. The first row lists the paper's title, the second row the keyphrases assigned by the author(s), whereas the third row contains the top five keyphrases extracted by KpEM: We highlight in bold style the "correct result," that is, in this case, the keyphrases that matched with the author-assigned keyphrases. For example, for the first paper, they are four: concurrent systems, model analysis, deadlock, and mdd.

Figure 6 synthesizes the results of the experiment, when 3, 5, 7, and 10 tags are suggested. The x-axis indicates the number of tags (keyphrases) recommended by PIRATES using the KpEM module, whereas the y-axis contains the average number of tags that matched with the keyphrases inserted by the author. The upper line in Figure 6 describes the behavior of an abstract, optimal recommender φ, which generates the set of tag recommendations assigning greatest relevance to keyphrases preassigned by each author to their documents. Obviously, this is just a theoretical upper bound to the performance of a recommender. No real recommender can in general perfectly conform with this performance indicator. We can observe that the number of correct suggestions grows quite slowly when the optimal recommender suggests more than seven keyphrases: this is explained by the fact that in our data set very few authors assigned more than seven keyphrases to their documents. Moreover, there are keyphrases suggested by the optimal recommender that cannot

Table IV. Top Five Keyphrases Extracted by KpEM for Three Sample Documents.

Document Title	"A UML/SPT Model Analysis Methodology for Concurrent Systems Based on Genetic Algorithms."	"An MDA Framework Supporting OCL".	"Integration of UML Views using B Notations."
Keyphrases assigned by document author	MDD deadlocks model analysis concurrent systems genetic algorithms SPT UML	MDE OCL model transformation code-generation verification	UML class operation use case event B method
Keyphrases extracted by KpEM	uml design models **concurrent systems** **model analysis** **deadlock** **mdd**	template-based code generator **model transformations** **code generation** mda **verification**	communication between state-charts **use cases** **class operations** **uml** refinement

be extracted looking only the text (some authors assigned keyphrases which express more the author's tacit knowledge than the concepts directly expressed in the textual documents themselves).

The second line in Figure 6 proposes the results of a second abstract optimal recommender, which considers only author-assigned keyphrases that are explicitly

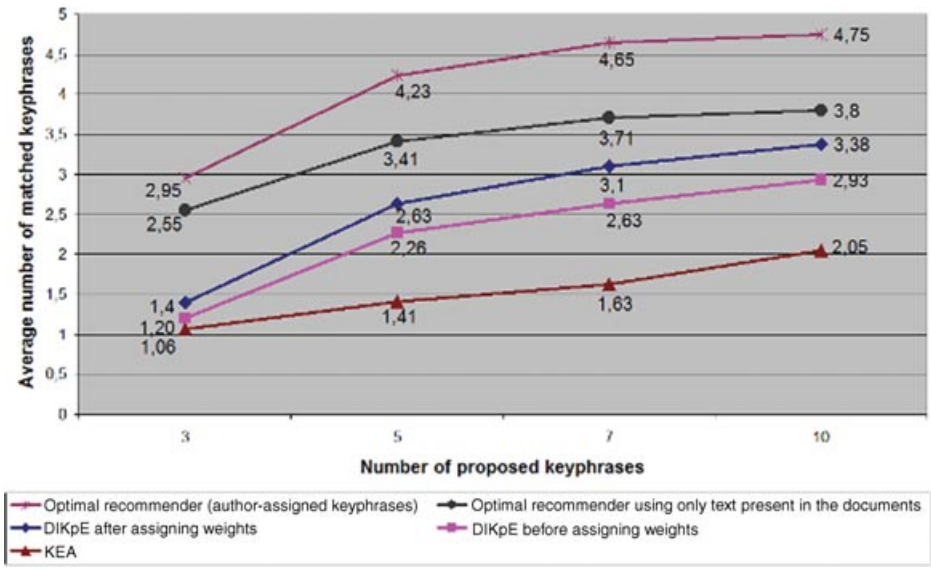

Figure 6. Overall performances.

stated in the document; this recommender "reasons in the same way as the documents' authors" without adding any form of knowledge, which is not explicitly stated in the documents. We can observe that there is a gap between the two hypothetical recommenders described in Figure 6. Obviously, the second line resembles more easily a real recommender and represents a more realistic upper-bound for any automatic tagging approach such as those of DIKpE and KEA. Nonetheless, it is interesting to compare both these hypothetical recommender systems with the performance of PIRATES, especially considering the role that a reasoning engine such as ORE can play to "simulate" any "intelligent" reasoning mechanism.

The last three lines in Figure 6 describe the performances of DIKpE after assigning weights, DIKpE before assigning weights and KEA. According to our evaluation criteria, DIKpE after assigning weights obtained better performance than the other two approaches where, however, DIKpE before assigning weights outperforms KEA. The Figure 6 shows, for example, that when five tags have been suggested, on average, DIKpE after assigning weights realized 2.63 matches, DIKpE before assigning weights 2.26 matches and KEA 1.41 matches.

To prove the strength of our results, some significance tests have been executed to verify whether the observed differences among KEA, DIKpE before assigning weights and DIKpE after assigning weights are truly meaningful or occurred by chance. The statistical tests want to disprove the null hypothesis that the compared approaches have the same performances and observed differences depend just on noise. To obtain this result, statistical tests have to calculate a p-value, that is the probability that observed differences occurred by chance. In particular, both a parametric test (the two tailed paired t-test), and a nonparametric test (the Wilcoxon test[60]) have been executed proving that DIKpE before assigning weights outperforms KEA in a statistically significant way (p-value ≤ 0.01) when 5, 7 or 10 tags are returned. Moreover, DIKpE after assigning weights always outperforms in a statistically significant way, DIKpE before assigning weights and KEA with a p-value ≤ 0.01. In conclusion, DIKpE after assigning weights provides the better results. Moreover, it is interesting to note also that, unlike KEA, our approach in keyphrase extraction is not affected by the performance degradation problem mentioned in Ref. 61, especially concerning the domain specificity.

6.3.2. ORE Tag Recommendations

Table-V contains the tags generated by ORE for the same three sample documents. All the tags recommended by ORE are to be considered as *new knowledge* automatically inferred by the system because they are not explicitly stated in the documents. In Table V, the second row lists the top 10 keyphrases extracted by KpEM; the row named "One-to-one ontology mappings" contains the terms, present in the ontology, which matched with the keyphrases provided by KpEM. If a keyphrase realizes a match with either an instance or a class present in the ontology, then the One-to-one term provided by ORE is the instance's class; otherwise, if the keyphrase matches with a class, ORE returns the immediate parent class in the hierarchy path. Finally, the row "Common ancestor" is computed by ORE by following the navigation strategy explained in Section 4.3.

Table V. ORE Tags for the Three Sample Papers Shown in Table-IV.

Document Title	"A UML/SPT Model Analysis Methodology for Concurrent Systems Based on Genetic Algorithms."	"An MDA Framework Supporting OCL".	"Integration of UML Views using B Notations."
KpEM keyphrases	**uml design models** types of concurrency concurrency problems concurrent systems model analysis **design models** deadlock **mdd** **uml** spt	**template-based code generator** functional programming language model repository **model transformations** **code generation** mda framework **mda** **verification** **ocl** uml/ocl	communication between state-charts class operation dependency abstract machine **use cases** **class operations** formal specification **uml** refinement concepts translation
ORE tags:	object-oriented_design	understandable_source_code_techniques	elicitation_techniques
One-to-one ontology mappings	object-oriented_design software_design_methodology software_design_notations	software_design_notations coding software_design_methodology software_testing formal_specification_language	aggregation_models software_design_notations
ORE tags:	software_design_strategies_and_methods	software_construction	software_requirements
Common ancestors	software_design	software_design	

The automated evaluation procedure utilized for assessing the quality of the KpEM tag recommendation is not suitable to properly measure the performance of ORE because it does not take into account any form of knowledge that is not presented in the documents themselves. Thus, the results have been manually validated by human experts in the field of software engineering. Even in the absence of an automated evaluation procedure, the approach used by ORE seems appropriate to integrate the type of knowledge used by other PIRATES modules and expressed by tags. Our preliminary results show how ORE is capable to infer new semantic-related knowledge that cannot be inferred looking only the textual content of the documents. This is clearly a challenging task. Looking at the results sampled in Table-V, the ORE preliminary performance seems to be promising. To completely assess the quality of the ORE tag recommendations, an experimentation involving a larger data set and more participants (groups of both students and experts in the field of software engineering) is in progress.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented an innovative and hybrid approach for automatically recommending content-based tags using keyphrases and ontologies. The keyphrase extraction works on a single document without any previous parameter tuning; a navigation strategy on ontology identifies meaningful ancestors for relevant extracted keyphrases and recommends them as new possible tags.

Further work will focus on the evaluation procedure. Relatively to KpEM, we assumed here that a keyphrase extraction system is optimal, if it provides the same keyphrases that an author defines for his (her) documents. However, in general there may exist many other keyphrases (different than those preassigned by authors) that are also appropriate for summarizing a given document. Relatively to ORE, we are working to involve a group of human experts to assess the quality of keyphrases generated using the ontology. Another aspect to consider is that domain ontologies are generally difficult to obtain. To limit this problem, we are planning to exploit publicly available search services tailored to automatically find ontologies, such as Swoogle,[m] OntoSearch,[n] SHOE,[o] or ECHOS.[p]

Thus, a further aspect to consider is to take into account the human subjectivity in assigning keyphrases, considering also adaptive personalization techniques for tuning the extraction process to the specific user's interests.

Finally, for the future work, we plan to investigate different ways to compute the coefficients of linear combination of features. We also need to concentrate on a better way to decide the number of keyphrases to be extracted by the system, instead of using a fixed number.

Summarizing our future research will be addressed on three main objectives:

- emphasize the aspects related to the personalization of the recommending process;
- generalize the proposed tag recommendation approach, to automate the extension of an ontology on the basis of user choices and preferences;
- refine the evaluation procedure, executing several real-life experiments involving volunteers in order to include in our evaluation also the human subjectivity to better measure the effective performance of the recommender system;
- investigate different ways to assign opportune parameters for the keyphrase extraction module, related, for example, to the coefficients to be used for the linear combination of features, and to the number of keyphrases to be extracted.

Acknowledgments

The authors acknowledge the financial support of the Italian Ministry of Education, University and Research (MIUR) within the FIRB project number RBIN04M8S8.

[m] http://swoogle.umbc.edu/
[n] http://www.ontosearch.org/
[o] http://www.cs.umd.edu/projects/plus/SHOE/search/
[p] http://www.ekoss.org/

References

1. Dattolo A, Duca S, Tomasi F, Vitali F. Towards disambiguating social tagging systems. Handbook of research on Web 2.0, 3.0 and X.0: Technologies, business and social applications. Hershey, PA: IGI-Global; 2010. Vol. 1, Ch. 20, pp 349–369.
2. Golder SA, Huberman BA. Usage patterns of collaborative tagging systems. J Inform Sci 2006;32(2):198–208.
3. Baruzzo A, Dattolo A, Pudota N, Tasso C. Recommending new tags using domain-ontologies. In: IEEE/WIC/ACM Int Conf on Web Intelligence and Intelligent Agent Technology. Milan, Italy; 2009. pp 409–412.
4. Musto C, Narducci F, Gemmis MD, Lops P, Semeraro G. Star: a social tag recommender system. In: Eisterlehner F, Hotho A, Jäschke R, editors. ECML PKDD discovery challenge. Bled, Slovenia: CEUR; 2009. Vol. 497, pp 215–227.
5. Heymann P, Ramage D, Molina HG, Social tag prediction. In: ACM SIGIR Int Conf on Research and Development in Information Retrieval. New York: ACM; 2008. pp 531–538.
6. Frank E, Paynter GW, Witten IH, Gutwin C, Nevill-Manning CG. Domain-specific keyphrase extraction. In: International Joint Conf on artificial intelligence. San Francisco, CA: Morgan Kaufmann; 1999. pp 668–673.
7. Hammouda KM, Matute DN, Kamel MS. Corephrase: keyphrase extraction for document clustering. In: Perner P, Imiya A, editors. Machine learning and data mining in pattern recognition, Lecture Notes in Computer Science. Berlin: Springer; 2005. Vol. 3587. pp 265–274.
8. Han J, Kim T, Choi J. Web document clustering by using automatic keyphrase extraction. In: IEEE/WIC/ACM Int Conf on Web Intelligence and Intelligent Agent Technology. Washington, DC: IEEE Computer Society; 2007. pp 56–59.
9. Krulwich B, Burkey C. Learning user information interests through the extraction of semantically significant phrases. In: Hearst M, Hirsh H, editors. Spring Symp on Machine Learning in Information Access. Stanford, CA: AAAI Press; 1996. pp 110–112.
10. Vibhu AB, Berger AL, Mittal VO. Ocelot: a system for summarizing Web pages. In: ACM SIGIR Int Conf on Research and Development in Information Retrieval. New York: ACM; 2000. pp 144–151.
11. D'avanzo E, Magnini B. A keyphrase-based approach to summarization: the LAKE system. In: Document Understanding Conference. Vancouver, Canada; 2005.
12. Wang J, Peng H, Hu JS. Automatic keyphrases extraction from document using neural network. In: Advances in machine learning and cybernetics, Lecture Notes in Computer Science. Berlin: Springer; 2005. Vol. 3930. pp 633–641.
13. Kosovac B, Vanier DJ, Froese TM. Use of keyphrase extraction software for creation of an aec/fm thesaurus. Electr J Info Technol Construction 2000;5:25–36.
14. Wu JL, Agogino AM. Automating keyphrase extraction with multi-objective genetic algorithms. In: Hawaii Int Conf on System Sciences. Washington, DC: IEEE Computer Society. 2004. pp 401–403.
15. Wu YFB, LiQ. Document keyphrases as subject metadata: incorporating document key concepts in search results. J Infor Retrieval 2008;11(3):229–249.
16. Song M, Song IY, Allen RB, Obradovic Z. Keyphrase extraction-based query expansion in digital libraries. In: ACM/IEEE-CS Joint Conf on Digital Libraries. New York: ACM; 2006. pp 202–209.
17. Zhou Z, Tian Y, Li Y, Liu T, Huang T, Gao W. PKU at imageclef 2008: experiments with query extension techniques for text-base and content-based image retrieval. In: Online Working Notes for the Cross-Language Evaluation Forum. Aarhus, Denmark: Springer; 2008.
18. Medelyan O, Frank E, Witten IH. Human-competitive tagging using automatic keyphrase extraction. In: Conf on Empirical Methods in Natural Language Processing. Morristown, NJ: Association for Computational Linguistics; 2009. pp 1318–1327.

19. Turney PD. Learning to extract keyphrases from text. National Research Council, Institute for Information Technology, Technical Report ERB-1057; 1999. Available online: http://www.extractor.com/ERB-1057.pdf (accessed April 19, 2010).
20. Turney PD. Learning algorithms for keyphrase extraction. J Inform Retrieval 2000;2:303–336.
21. Turney PD. Coherent keyphrase extraction via Web mining. In: Int Joint Conf on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann; 2003. pp 434–439.
22. Witten IH, Paynter GW, Frank E, Gutwin C, Manning CGN. Kea: practical automatic keyphrase extraction. In: ACM Conf on Digital Libraries. New York: ACM; 1999. pp 254–255.
23. Chen M, Sun JT, Zeng HJ, Lam KY. A practical system of keyphrase extraction for Web pages. In: ACM Int Conf on Information and Knowledge Management. New York: ACM; 2005. pp 277–278.
24. Hulth A. Improved automatic keyword extraction given more linguistic knowledge. In: Empirical methods in natural language processing. Morristown, NJ: Association for Computational Linguistics; 2003. pp 216–223.
25. Avanzo E, Magnini B, Vallin A. Keyphrase extraction for summarization purposes: the LAKE system at DUC2004. In: Human Language Technology Conf / North American Chapter of the Association for Computational Linguistics Annual Meeting, Boston, MA; 2004.
26. Wang C, Li S, Wang W. Experiment research on feature selection and learning method in keyphrase extraction. In: Int Conf on Computer Processing of Oriental Languages. Language Technology for the Knowledge-Based Economy. Berlin, Germany: Springer-Verlag; 2009. pp 305–312.
27. Barker K, Cornacchia N. Using noun phrase heads to extract document keyphrases. In: Biennial Conf of the Canadian society on Computational Studies of Intelligence. London: Springer-Verlag; 2000. pp 40–52.
28. Bracewell DB, Ren F, Kuroiwa S. Multilingual single document keyword extraction for information retrieval. In: IEEE Int Conf on Natural Language Processing and Knowledge Engineering. Wuhan, China; 2005. pp 517–522.
29. Liu Z, Li P, Zheng Y, Sun M. Clustering to find exemplar terms for keyphrase extraction. In: Int Conf on Empirical Methods in Natural Language Processing. Singapore: Association for Computational Linguistics; 2009. pp 257–266.
30. Wan X, Xiao J. Single document keyphrase extraction using neighborhood knowledge. In: National Conf on Artificial Intelligence. AAAI Press; 2008. pp 855–860.
31. Mihalcea R, Tarau P. Textrank: bringing order into texts. In: Int Conf on Empirical Methods in Natural Language Processing. Barcelona, Spain: ACL; 2004. pp 404–411.
32. Litvak M, Last M. Graph-based keyword extraction for single-document summarization. In: Multi-Source Multilingual Information Extraction and Summarization. Morristown, NJ: Association for Computational Linguistics; 2008. pp 17–24.
33. Huang C, Tian Y, Zhou Z, Ling CX, Huang T. Keyphrase extraction using semantic networks structure analysis. In: Int Conf on Data Mining. Washington, DC: IEEE Computer Society; 2006. pp 275–284.
34. Brin S, Page L. The anatomy of a large-scale hyper textual web search engine. J Comput Networks ISDN Syst 1998;30(1–7):107–117.
35. Sotomayor B. Folk2onto: mapping social tags into ontological categories. University of Chicago, Technical Report; 2006. Available online: www.deli.deusto.es/Resources/Documents/folk2onto.pdf (accessed April 25, 2010).
36. Tenenbaum L, Shapira B, Shoval P. Ontology-based classification of news in an electronic newspaper. In: INFOS. Varna, Bulgaria; 2008. pp 89–97.
37. Acampora G, Gaeta M, Loia V. Exploring e-learning knowledge through ontological memetic agents. IEEE Comput Intell Maga 2010;5(2): 66–77.
38. Adamo A, Cafaro L, Loia V, Romano C, Veniero M. A multi-layered agent ontology system for resource inventory. In: IEEE Int Symp on Industrial Electronics. Cambridge, UK; 2008. pp 2317–2322.

39. Mishne G. Autotag: a collaborative approach to automated tag assignment for Weblog posts. In: Int Conf on World Wide Web. New York: ACM; 2006. pp 953–954.
40. Tatu M, Srikanth M. Silva TD. Rsdc'08: tag recommendations using bookmark content. In: ECML PKDD Discovery Challenge. 2008. pp 96–107.
41. Baruzzo A, Dattolo A, Pudota N, Tasso C. A general framework for personalized text classification and annotation. In: Int Conf on User Modelling and Personalization. Trento, Italy; 2009. pp 31–39.
42. Asnicar F, Tasso C. Ifweb: a prototype of user model-based intelligent agent for document filtering and navigation in the World Wide Web. In: Int Conf on User Modelling. Chia Laguna, Sardinia; 1997.
43. Casoto P, Dattolo A, Ferrara F, Pudota N, Omero P, Tasso C. Generating and sharing personal information spaces. In: ACM Int Conf on Adaptive Hypermedia and Adaptive Web-Based Systems. Hannover, Germany; 2008. pp 14–23.
44. Baruzzo A, Casoto P, Dattolo A, Tasso C. Handling evolution in digital libraries. In: Italian Res Conf on Digital Library Management Systems. Padova, Italy; 2009. pp 34–50.
45. Pudota N, Casoto P, Dattolo A, Omero P, Tasso C. Towards bridging the gap between personalization and information extraction. In: Agosti M, Esposito F, Thanos C, editors. Italian Res Conf on Digital Library Management Systems. Padova, Italy; 2008. pp 33–40.
46. Porter MF. An algorithm for suffix stripping. In: Readings in information retrieval. San Francisco:Morgan Kaufmann; 1997. pp 313–316.
47. Justeson J, Katz S. Technical terminology: some linguistic properties and an algorithm for identification in text. J Nat Lang Eng 1995;1:9–27.
48. Hulth A, Megyesi BB. A study on automatically extracted keywords in text categorization. In: Int Conf on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics. Morristown, NJ: Association for Computational Linguistics; 2006. pp 537–544.
49. Ercan G, Cicekli I. Using lexical chains for keyword extraction. J Inform Process Manage 2007;43(6):1705–1714.
50. Nguyen TD, Kan MY. Keyphrase extraction in scientific publications. In: Goh DHL, Cao TH, ølvberg IS, Rasmussen EM, editors. Lecture Notes in Computer Science. Berlin: Springer; 2007. Vol. 4822. pp 317–326.
51. Kumar N, Srinathan K. Automatic keyphrase extraction from scientific documents using *n*-gram filtration technique. In: ACM Symp on Document Engineering. New York: ACM; 2008. pp 199–208.
52. Quillian M. Semantic memory. In: Minsky M, editor. Semantic information processing. Cambridge, MA: MIT Press; 1968. pp 227–270.
53. Baruzzo A, Casoto P, Challapalli P, Dattolo A, Pudota N, Tasso C. Toward semantic digital libraries: exploiting Web 2.0 and semantic services in cultural heritage. J Digital Inform Berlin; 2009;6(10). Available at: http://journals.tdl.org/jodi/article/viewarticle/688/576 (accessed September 29, 2010).
54. Kruk S, McDaniel B. Semantic digital libraries. Berlin: Springer Verlag; 2009. 246 p.
55. Eide O, Felicetti A, Ore C, Andrea AD, Holmen J. Encoding cultural heritage information for the semantic Web. Procedures for data integration through CIDOC-CRM mapping. In: Arnold D, Niccolucci F, Pletinckx D, Gool LV, editors. EPOCH conference on open digital cultural heritage systems. Rome, Italy; 2008. pp 1–7.
56. Hernandez F, Rodrigo L, Contreras J, Carbone F. Building a cultural heritage ontology for cantabria. In: CIDOC Annual Conf. Athens; 2008.
57. Kruk SR, Woroniecki T, Gzella A, Dabrowski M. Jeromedl—a semantic digital library. In: Golbeck J, Mika P, editors. Semantic Web challenge. Busan, South Korea: CEUR; 2007. Vol. 295. pp 139–150.
58. Sluijs KVD, Houben GH. Metadata-based access to cultural heritage collections: the RHCe use case. In: Int Conf on Adaptive Hypermedia and Adaptive Web-Based Systems. Hannover, Germany; 2008. pp 15–25.

59. Hunter J, Khan I, Gerber A. Harvana: harvesting community tags to enrich collection metadata. In: ACM/IEEE-CS Joint Conf on Digital Libraries. New York: ACM; 2008. pp 147–156.
60. Hull D. Using statistical testing in the evaluation of retrieval experiments. In: ACM SIGIR Int Conf on Research and Development in Information Retrieval. New York: ACM; 1993. pp 329–338.
61. Jones S, Paynter GW. Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications. J Am Soc for Inform Sci Technol 2002; 53(8):653–677.