Multimedia Tools and Applications (2020) 79:7577–7598
https://doi.org/10.1007/s11042-019-08252-2

An ontology-driven multimedia focused crawler based on linked open data and deep learning techniques

Andrea Capuano[1] · **Antonio M. Rinaldi**[1,2] · **Cristiano Russo**[1]

Received: 23 February 2019 / Revised: 31 July 2019 / Accepted: 13 September 2019 /
Published online: 24 December 2019

Abstract
Web-page indexing and classification have been studied extensively starting from the early WWW years. A smart intelligent web agent called focused crawler is a specific software able to seek web pages that are relevant to a particular topic domain. In this article we propose a novel approach to focused crawling based on the use of both textual and multimedia web page content. In our approach we define a novel strategy to choose if a web page should be further explored. We implement our framework in a system which aims to improve the crawling task using semantic based techniques and combining the results with novel technologies like convolutional neural networks and linked open data. Our framework uses ontologies to correlate different topics and understanding their relationships. The correlation among topics is used to improve a textual topic detection step. These results are combined with multimedia analysis and classification based on convolutional neural networks to extract image features. Experimental results are also presented and discussed in order to measure the effectiveness of our framework compared with other approaches using a ground truth composed of web pages about a specific domain.

Keywords Focused crawling · Knowledge engineering · Document analysis · Multimedia processing · Ontologies · Document classification · Convolutional neural networks · Linked open data

✉ Antonio M. Rinaldi
antoniomaria.rinaldi@unina.it

Andrea Capuano
and.capuano@gmail.com

Cristiano Russo
cristiano.russo@studenti.unina.it

[1] Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Napoli, Italy

[2] IKNOS-LAB Intelligent and Knowledge Systems (LUPT), Napoli, Italy

1 Introduction

In the last years, the size of websites has grown drastically through the addition of different data formats in particular multimedia components. According to recent analysis of the Web structure and its content, the size of the indexed web contains at least 4.49 billion pages [21], and the deep web is considered to be 500 times larger than the surface web [11]. In this context the use of automatic agents to explore and index such large collection of documents is a crucial task in the whole web information retrieval process.

The indexing of large collections is a challenging task and for this reason automatic agents like web crawlers have been designed and implemented since the beginning of the web age. Moreover, other techniques used for document classification has been investigated to improve the quality of the web crawlers. In the web page analysis it is possible to perform the classification step after its indexing but some information retrieval tasks need to seek web page according to predefined criteria and fetch only documents which satisfy constraints related to a specific topic.

The task of crawling web page according to topical relatedness is tightly related to document classification and the relative web agent is called focused crawler [18].

Web-pages classification, also known as web page categorization or subject classification, is a subset of the text classification problem that has been formally defined by [52]. Different techniques have been proposed and implemented during the years based both on textual and multimedia data [5, 7, 33, 47].

In the last years deep neural networks (DNN) have been extensively investigated due to the high performance accuracy in image analysis. Different deep neural network architectures have been proposed [36, 51] and in particular convolutional neural networks (CNN) provide the best performances in image classification and object recognition tasks. We use CNN in order to consider web page multimedia contents to improve the classification task.

The aim of this article is to provide a complete framework for focused crawling process combining novel techniques like ontology-driven topic detection, linked open data (LOD) and deep learning technologies.

In our approach we use ontologies to represent our domain of interest and implement the support structure to our algorithms and techniques. Using this cognitive tool we can also organize categories in a more effective way taking into account their relationships. Moreover, an innovative use of formal and not-user oriented ontologies during the crawling process allows a high level of generalization following a novel crawling strategy. The whole framework has been implemented in a system to crawl web pages about specific conceptual domains.

We try to give a new solution to the issue related to the use of different features to identify specific web pages to crawl and to explore in a novel and effective way the related web graph.

The proposed strategy has been compared with some standard techniques presented in literature using a ground truth composed of web pages about the animal domain and obtained results show important improvements.

The presented framework for focused crawling takes advantage of the above cited techniques and propose novel strategies and algorithms to explore portions of the web in a more effective way and categorize contents. The implemented crawler fetches web pages starting from a seed URL and categorizes them according to a novel combination of textual and multimedia features.

The article is organized as follow: in Section 2 a literature review is presented and discussed putting in evidence the novelties of our approach; the system architecture and the proposed methodology for document annotation is discussed in Section 3; a use case of our crawler together with experimental results are in Section 4; eventually, conclusions and future works are presented in Section 5.

2 Related work

Web crawlers can be divided into general-purpose and special-purpose web agents; the last ones are also known as focused crawlers [42]. In this section we consider several relevant works related to focused crawlers leaving out a literature review on general-purpose category. For the sake of clarity, we highlight a distinction between the existing approaches. In particular, since our framework is based on ontologies, we consider works using this approach and, on the other hand, we provide a review of other techniques putting in evidence the effectiveness of CNN features used as generic features extractors for an image retrieval task.

A focused crawler is a robust agent with specific abilities in avoiding large perturbations in the starting set of URLs discovering large overlapping sets of resources. It is capable to explort and discovert valuable resources far from the starting set and efficiently pruning millions of pages out of its domain of interest.

Trained focused crawlers apply machine learning techniques to perform their tasks. Online taxonomies or manually categorized datasets can be used as training set for supervised focused crawling. Some algorithms that seem to be more successful in this context are based on support vector machines (SVM) and neural networks (NN) [44].

From a general point of view, the pioneering work about focused crawlers is [18] in which the authors designed two hypertext mining software that guide the crawler: (i) a classifier to evaluate the relevance of a hypertext document with respect to the focus topics, and (ii) a *distiller* to recognize hypertext nodes that present a number of access points to many relevant pages within few links.

Genetic algorithms also provide interesting performances in the task of focused crawling. Yohanes et al. [57] argue that genetic algorithms achieve high harvest ratio and avoid problems of local searching algorithms. The fitness function assigned to a page is obtained using Jaccard's similarity based on the ratio of the number of intersection links and union links between two web pages considering as basic assumption that heavily connected pages are also topically related. Several studies about text analysis using similar approaches are in [2–4, 39].

Diligenti et al. [23] present a focused crawler based on context graph, decision tree and DOM. Given a topic of interest, a context graph is used to provide a list of categories related to the topic. The use of DOM has been also used in [19] to order the URLs in a web-page according to a relevance metric or ordering scheme. Several metrics have been proposed by the author as driving query similarity, backlink count, pagerank and location metrics.

In [55] the authors propose an improved retrieval model, the Semantic Similarity Vector Space Model (SSVSM), which integrates the TF*IDF values of terms and the semantic similarities among terms. This model has been used to build topic and document semantic vectors that are mapped into the same double-term set, and computes the cosine similarities between these semantic vectors as topic-relevant similarities of documents, including the full texts and anchor texts of unvisited hyperlinks. In this way, the model can be used to predict the priorities of unvisited hyperlinks integrating full text and anchor text topic-relevant similarities.

Batsakis et al. [10] propose a novel learning crawler based on Hidden Markov Model (HMM). This kind of crawler is capable to learn not only the content of relevant pages but also paths in the relevant pages.

Ontologies are also used as enabling technology for focused crawling. In our work we consider an ontology as an explicit and formal specification of a shared conceptualization [28].

An early study on the use of an ontology-based focused crawler is presented in [25]. The authors propose a two cycle focused crawler. The ontology cycle is implemented with a user provided ontology, while the crawling cycle is devoted to the retrieval of documents considered to be relevant. A relevance score is computed by a metric considering the input document and the provided ontology.

A similar approach is proposed by Kozanindis [34] where ontologies are used to effectively identify web pages related to pre-defined topics. After the web page fetching and pre-processing steps, a thematic keyword extraction is implemented to compute a topic relevance scoring based on a domain ontology.

In [40] the authors propose a weight table of ontology terms created by domain experts following the same approach previously cited.

OntoDir [45] is an ontology based crawler to categorize web page in according to a semantic relatedness metric and a categorization schema based on a formal knowledge base.

A topic ontology is used in [53] for the focused crawler pre-processing task providing a semantically expanded search to improve its effectiveness.

Yang [56] proposes Ontocrawler, a focused crawler which makes an extensive use of ontologies to build website models that effectively support web search. These models are built using a component called OntoAnnotator able to annotate ontology terms in the web page. Web sites models are then sent to a classifier that compute relevance of the web page.

In [29] the author propose an unsupervised adaptive ontology-learning process to enhance performance of a focused crawler. Furthermore, they adopt approaches such as large scale crawling and they use hadoop to support big data together with semantic web technologies.

In our work we are also interested in the use of image analysis to improve the quality of our focused crawler in the processing of web page contents.

Content based image retrieval (CBIR) is a fervent research field and with the extended study and use of convolutional neural networks several novel techniques to support the features extraction task have been explored and proposed in recent years.

Oquab et al [43] propose an approach to learn and transfer mid-level image features extracted from a CNN and use them for different visual recognition tasks.

A general approach is also proposed by [24] where a framework called DeCAF is proposed to extract features from a CNN. The authors show how features extracted from a CNN and trained for a given task generalize well on an unrelated task. They also demonstrate the generality and semantic knowledge implicit in these features.

In [48] the authors investigate how features extracted from the overfeat convolutional neural networks can be used for image retrieval, in particular they compare it to state-of-the-art pipelines such as VLAD and BoVW. The authors propose of dividing images in sub-patches and feed-forward them to a CNN and compare through L2 distance the query image to the reference images.

A significant enhancement in the use of CNN in content based image retrieval is shown by [30] where the authors explore how content based image retrieval can be improved by using features extracted from a CNN. The authors propose three schemes and among them, they propose the extraction of activation features from the fully connected layers of a CNN using a simple feed-forward pass of an image through the network. The authors explore how well these models work on some tasks that are unrelated to the training tasks of the CNN and eventually explain how these approaches can fill the semantic gap that exists between low-level image features and semantic concepts perceived by humans.

Other works [27, 31, 49] confirm the same intuition from the aforementioned articles and explain how CNN feature representation can be used as a generic representation even for unsupervised recognition tasks and in particular for image retrieval. Other interesting improvements have been obtained from convolutional neural network for content-based image retrieval by [8, 31, 59] using features extracted from pooling layers and used in image retrieval tasks.

The proposed framework introduces several novelties with respect to the discussed literature based on the used of a formal non-user oriented ontology, CNN-based multimedia retrieval and linked open data. We combine the results obtained by the LOD-based textual classification and the CNN-based multimedia retrieval to produce a classification. Moreover, we take advantage from the formal ontology to ensure that the classified web page is in line with the chosen domain. In addition, a novel strategy used to explore the web graph related to the recognized web pages starting to seed documents hase been proposed and tested to show the advances of the proposed framework with respect to standard techniques.

3 The proposed framework

A sketch of the proposed system architecture is shown in Fig. 1. Our implemented system is composed by several modules to support all the focused crawler tasks. The crawling process starts from an initial seed provided by an URL or a word. The system has a graphical user interface which is composed by a text box used to define the seed. A conceptual domain has to be provided as an ontology, which will be imported in an instance of the Neo4j graphDB due to the characteristic of this kind of NoSQL database [14]. Using these information the crawler can start the web exploration.

Starting from the initial seed, the main crawler controller creates the crawler threads scheduling the tasks. The crawler threads receive as input an URL from the scheduler, fetch the web page and collect the outgoing links.

Once the web page is fetched, it is sent to the pre-processing block which is in charge of analyzing the page structure and extract text and images. Text is further cleaned removing

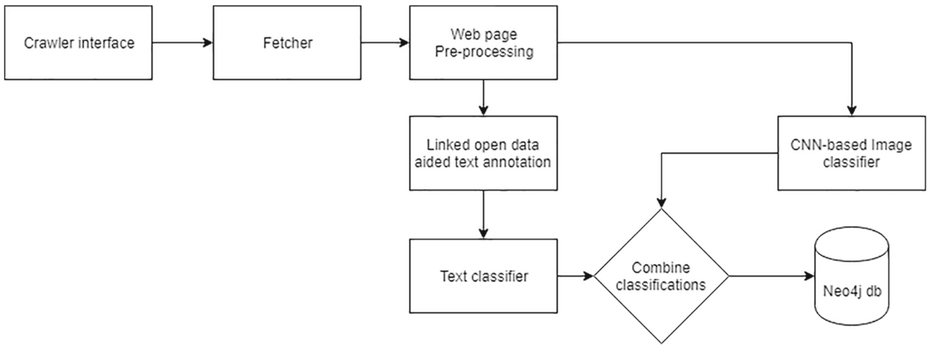

Fig. 1 System architecture overview

stopwords to enhance the successive steps. The text classification block uses several heterogeneous technologies to perform a prediction about the relevance of a web page within a selected conceptual domain. If a page is relevant, a candidate main topic is also selected.

Linked Open Data (i.e.DBpedia and wikidata) are used to analyze the web page text content. We take advantage of these technologies because they provide an efficient way to annotate web page raw text and find entities that are semantically related between each other. These entities are well defined and have specific properties that can be used to infer a category for a given web page.

The CNN-based image-classifier computes a similarity score between the downloaded images and the ones stored in the knowledge base. A a multimedia content web page classification is inferred from those similarity scores.

Textual and image classification are then combined to classify the web page. Once a main topic is found, we recognize this topic in the domain ontology classes to understand if the classified web page is relevant to the chosen conceptual domain. If the page is relevant, it is inserted in the Neo4j database instance, and linked to the corresponding node representing the detected topic.

The Neo4j instance has been initially populated with data from ImageNet [22] following a strategy described in [12, 13]. ImageNet is a database of images organized according to the hierarchy of WordNet [38] (currently only for nouns) in which each node is represented by thousands of images with an average of about 500 images per node. The conceptual domain for the focused crawler is a slice of ImageNet regarding *"dogs"*. We choose this domain due to the high number of textual and visual information about it and the wide availability of web documents related to this topic. Moreover, the term *dog* has various meanings belonging to very different conceptual domains. These characteristics allow us to have an interesting and challenging context of application for testing our framework efficacy.

The proposed system has been designed and implemented with a full modular approach and the different components will be described. In Fig. 2 a detailed view of the crawler architecture is shown. The main components are highlighted with dashed rectangles.

The crawling control is in charge of manage the crawling process, the pre-processing block transforms raw web pages in a simplified form using textual transformations and store the fetched images. The text classification component uses the text extracted from the web page to produce a classification about the web page topic. The CNN-based image classifier is implemented with a deep neural network to extract features from web page images and compute a similarity scores that will used to infer a topic about the web page. The combination block performs the final classification merging the results of textual and image classifier.

3.1 Crawling control

The crawler control block is the core component for the spidering and fetching subatsks. It is initialized through the graphical user interface, which also provides the following parameters to set for having a correct system initialization:

- *Initial seed*: this information is used to start the crawling process. It can be an URL or a word; in the first case it is a web address, while in the second case the system uses results from a search engine (i.e. Google) to provide an initial URL;
- *Number of crawler threads*: this parameter sets the number of threads to speed up the crawling process. It is set according to the hardware resources;

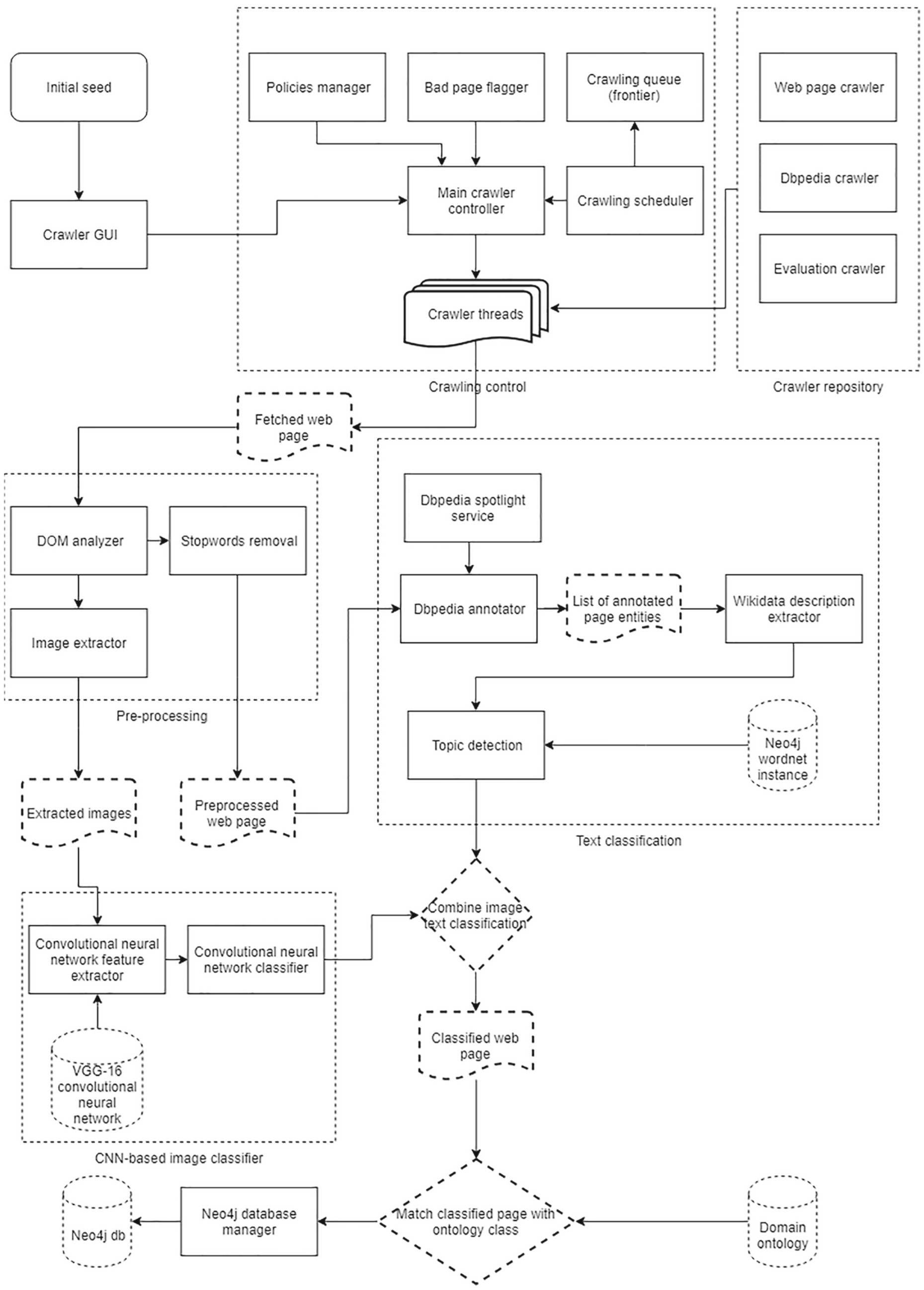

Fig. 2 Focused Crawler components

– *Type of crawler*: three types of crawlers are available: (i) Web page crawler - This crawler is specialized in crawling web page;DBpedia crawler - This agent is specialized in DBpedia crawling and it is used to populate the Neo4j database with instances of the ontology classes; (iii) Evaluation crawler - It is mainly used to fetch web pages from a controlled dataset to allow the system evaluation;

Once the crawling controller is initialized the seed page is fetched and its outgoing links are sent to the scheduler. The main scheduler will be in charge of assigning web page addresses among threads.

Three policies have been used in our crawler: the selection policy, the re-visit policy and the politeness policy [9]. Focused crawling itself is a specific type of selection policy then it is set by the chosen topic domain. Our crawler has been built to be a polite agent implementing three basic rules:

- *Crawler identification*: This is done by setting the user-agent; the defined user agent is "Dogs focused crawler".
- *Compliance with robots.txt (Robot exclusion protocol)*: We use crawler4j library to define and implement all common rules of the robots exclusion protocol.
- *Low bandwidth usage on a given web site*: The crawler is configured with a maximum-requests-per-second parameter. This means that given a particular web domain, the crawler will avoid flooding it with requests and observe a fixed time-constraint. This parameter can be changed by a configuration file during the crawling bootstrap.

The analyzed outlinks from a given web page could be not relevant considering the crawler domain of interest. The implemented system takes into account this issue and if more than a certain percentage of the outgoing links of a given web page are considered to be not relevant an early stopping criteria of the crawling process for the outlinks of the flagged page is performed.

In Fig. 3 the flagger process is shown. When a new page is ready to be visited, the system verifies that the source URL, i.e. the page whose outlink was the page we are currently visiting, is not a flagged page.

In case it was flagged, the page is not visited, otherwise it is analyzed. If the processed page is considered not relevant to the domain of interest, a counter is increased and a ratio is computed as follows:

$$\text{Percentage of irrelevant pages} = \frac{\text{Number of irrelevant pages}}{\text{Number of outlinks}}$$

The value of the ratio grows while the outlinks are explored and the pages are classified as irrelevant.

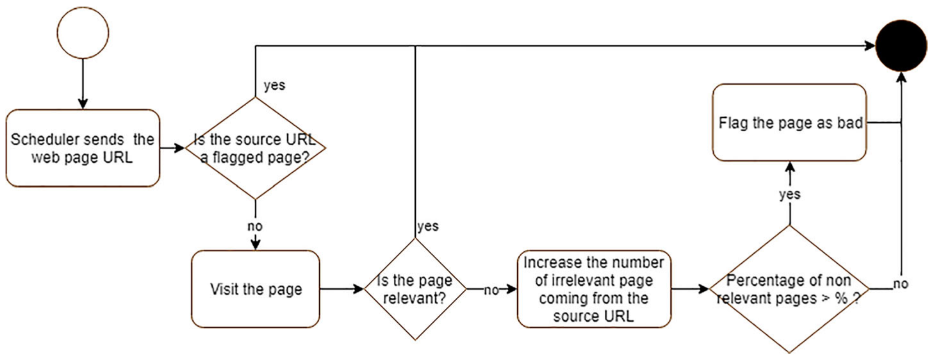
Scheduler sends the web page URL
Is the source URL a flagged page?
yes
no
Visit the page
Is the page relevant?
yes
no
Increase the number of irrelevant page coming from the source URL
Percentage of non relevant pages > % ?
yes
no
Flag the page as bad

Fig. 3 Bad page flagger flowchart

The pseudo-code of the crawler visiting algorithm is shown in Listing 1.

The overall crawling strategy is a combination of the crawler4j frontier scheduling algorithm, and the focused crawling selection policy. Crawler4j is the underlying java crawler engine used in the proposed framework; it allows the use of customizable priorities. A crawl frontier is a data structure that is used to store the URLs that will be processed and crawled. The basic operations for this data structure are adding, reordering and selecting URLs. The frontier is a specialized version of a priority queue.

The frontier of the proposed system will comply with the following rules:

– Outgoing links from the seed page will always be explored.
– Outgoing links from pages that were judged to be irrelevant will not be fetched.
– Outgoing links from pages that were judged to be relevant, but that were successively flagged bad will not be further visited.
– Links of DBpedia domain will be added to a special queue that will be processed by the DBpedia crawler.
– URLs will be dynamically selected from the crawler frontier according to a priority algorithm that is described in the following section.

The crawl frontier is a priority queue that is specialized for url-prioritizing tasks. An URL priority is a positive integer number and lower is its value higher is its the priority. If no priority is specified by a user, the frontier will follow a normal FIFO queue and the basic strategy will be a breadth-first exploration starting from the root. URL priority is assigned to the URL in the first visit and it will not be further changed.

In the proposed system the breadth-first search strategy (BFS) for web exploration is used since it has been proved to be very efficient for crawling tasks [41]. However, we added a rule to the BFS: *if a WWW URL belongs to a domain name that was previously found to be relevant it will be prioritized with respect to URLs whose domain names were previously explored and judged to be non relevant*. It is important to highlight the difference between a WWW domain name and an URL. For example given the URL: *"https://www.wikiart.org/en/caravaggio"* its domain is *"wikiart.org"*. The basic assumption is that web pages fetched from a domain whose previously crawled URLs were judged to be relevant have a higher chance of being relevant themselves. If the found domain was never explored in previous crawling sessions the system will use BFS.

3.2 Web page pre-processing

The pre-processing task is a crucial step for the text classification process. As a matter of fact for a generic web page, there are some parts of the text that are more important than others.

```
def should_visit(url):
    parent_url = url.get_parent()
    if(url == SEED_PAGE or parent_url == SEED_PAGE):
        return true

    parent_is_relevant = is_relevant(parent_url)
    parent_is_flagged = is_flagged(parent_url)
    if(parent_is_relevant and not parent_is_flagged):
        return true

return false
```

Listing 1 Crawler visiting algorithm

Web pages have a semi-structured nature, they are readable by both humans and automated agents. HTML is the most commonly used language on the web and it uses several tags to structure the web-page.

The DOM analyzer component extracts text only from the most content-rich parts of the document. The use of these tags can be very useful to extract rich context text [26].

Once the text is extracted, a stopwords removal step is performed. Stopwords are words that appear too frequently and represent noise in the text analysis because they don't have informative content [9]. The pre-processed text is then sent to the text-classification block.

The web-page is also mined for images fetching because the system will perform both textual and visual classification. A web-page usually consists of several images of different size, nevertheless, not all images are the same relevance for web page selection. Hence, too small size images (e.g. icons) will not be fetched. All the other images are then extracted and sent to the CNN-classifier block.

3.3 Web page classification

Once the page text has been extracted the next step is to classify web pages in according to the focused crawler selection policy. We point out that the classification process uses different informative sources as Linked Open Data and ontologies. It can be formalized as a function:

$$T \rightarrow f(A, D, C, O)$$

where T is the detected topic; A is a list of DBpedia annotations extracted from the text; D is a set of wikidata descriptions for the annotated entities; C represents a set of candidate topics for the given entities; O is a list of ontology classes.

We use two LODs repositories, i.e. DBpedia and wikidata, for our purposes. DBpedia is used to annotate entities within the page and to link these entities among each other. Moreover, we use wikidata to extract a brief and precise description of the annotated entities.

The pseudo-code for the topic detection is shown in Listing 2.

Once the text has been extracted it is passed to the text classification block. The DBpedia annotator interacts directly with the DBpedia spotlight service [37]. It is a tool for automatically annotating mentions of DBpedia resources in the text, providing a solution for linking unstructured information sources to the Linked Open Data cloud through DBpedia.

Given an unstructured text expressed in natural language, the annotator returns a list of DBpedia entities extracted from the document. DBpedia spotlight disambiguates the recognized terms and tries to choose the most suited DBpedia entity. The DBpedia spotlight APIs allow to select entities of a specific category, e.g. selecting only entities of type "Animal" given a raw text.

```
def topic_detection(extracted_text, ontology):
    candidate_entities = []
    entities =DBpedia_annotate(extracted_text)
    top_5_entities = count_and_sort(entities)
    for entity in top_5_entities:
        wikidata_description = get_wikidata_description(entity)
        detected_topic = topic_detection(wikidata_description)
        if detected_topic is in ontology:
        candidate_entities.append(entity)

    return candidate_entities
```

Listing 2 Topic detection alghoritm

Using this approach is possible to have a high degree of system reconfigurability from a conceptual domain point of view. In fact, if the DBpedia ontology and the focused crawler ontology have a good overlapping, the annotation step can give better results since it will not annotate entities that are outside from the specific ontological domain. This is possible using LOD technology.

The annotations extracted from a given text have to be analyzed. We follow the Luhn assumption and its application based on statistical information [9] but we use it on annotated entities and not for simple keywords. Therefore, we suppose that more frequent an entity is annotated and more important it is for a given text.

In the context of information retrieval the most frequent terms are often stopwords and, for this reason, the term frequency is usually used in combination with other schemes such as $TFxIDF$. In our case this problem is solved by DBpedia spotlight which doesn't annotate stopwords but only DBpedia entities. Moreover, stopwords are filtered from the original text during the preprocessing step.

Once the entities are sorted by frequency, the system consider the first K entities where K is a tunable parameter that can be set before initializing the crawling session. For each of these entries, related DBpedia pages are crawled.

The Fig. 4 shows the steps in the Web page annotation task.

It's important to point out that not all the annotated entities might exist at crawling time. If a missed entity is annotated, the crawler will create a Neo4j node that will be generated by the DBpedia page. This is an important task because if this kind of entity will be chosen as main-topic, the web page will have to be linked to the related Neo4j node.

The DBpedia page contains several information about the annotated entity. The properties (fields) of the entity are high dependent to the ontology class which this particular entity is assigned.

One of the most important issue is that all the properties are structured. The choice to use some of the proposed technologies in our framework is for example that one of the main efforts of DBpedia is to transform unstructured wikipedia data to structured one. This allows a great flexibility in the transformation of a DBpedia entity in a Neo4j node because a property mapping schema has been implemented using Neo4j schemaless property.

One of the most important bottlenecks in the crawling process is the computational overload due to the number of analyzed pages and their contents.

We try to smooth this problem using DBpedia entity linked data to wikidata entities. Wikidata is an open machine readable knowledge base similar to DBpedia but enriched with human created structured content on wikipedia documents. Wikidata provides a concise

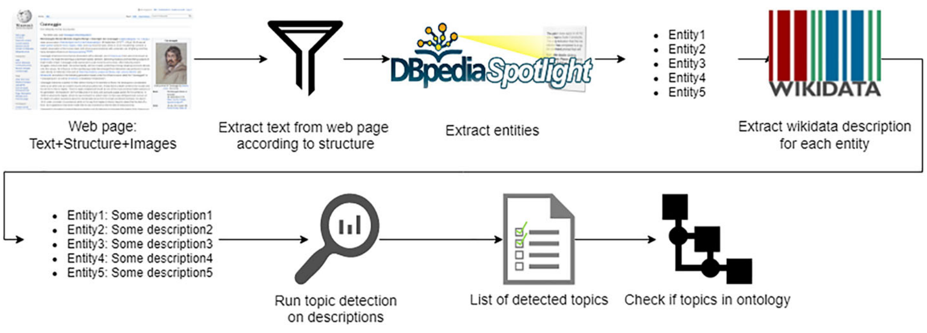
Web page:
Text+Structure+Images
Extract text from web page according to structure
DBpedia Spotlight
Extract entities
Entity1
Entity2
Entity3
Entity4
Entity5
WIKIDATA
Extract wikidata description for each entity
Entity1: Some description1
Entity2: Some description2
Entity3: Some description3
Entity4: Some description4
Entity5: Some description5
Run topic detection on descriptions
List of detected topics
Check if topics in ontology

Figure 4 Web page annotation pipeline

sentence for each item giving an accurate description of a given entity. This information simplify the crawler text classification task.

In this step of the the pipeline a list of entities with a brief descriptions is generated. These entities are possible candidates for the web page linking. However, it’s necessary to understand if such annotated entities are related to the specific domain of interest, for this reason the next step is to run a topic detection in order to get a list of candidate topics for a given web page. The textual component of our knowledge base (i.e. ImageNet) is derived from WordNet. Therefore we can use it for our metrics.

We propose a topic detection algorithm using a metric that combines two components namely, a syntactic semantic grade and a semantic grade:

$$TD(\nu) = SSG(\nu) + SeG(\nu) \tag{1}$$

where the SSG is a measure of the ambiguity of a term and is defined as:

$$SSG(\nu) = \sum_{i=1}^{n} \frac{1}{poly(w_i)} \tag{2}$$

where ν being the considered document, w_i being the analyzed word and $poly(w_i)$ is the number of meanings af a given term. This information is carried out form WordNet.

The SeG is based on the semantic distance between two WordNet concepts represented by terms. More specifically, it is based on a combination of the path length between pairs of terms and the depth of their subsumer, expressed as the number of hops from the roots of WordNet:

$$SeG(\nu) = \sum_{(w_i,w_j)} e^{-\alpha \cdot l(w_i,w_j)} \frac{e^{\beta \cdot d(w_i,w_j)} - e^{-\beta \cdot d(w_i,w_j)}}{e^{\beta \cdot d(w_i,w_j)} + e^{-\beta \cdot d(w_i,w_j)}} \tag{3}$$

where ν is the considered document, (w_i, w_j) being the pairs of words and $\alpha \geq 0$ and $\beta \geq 0$ are two scaling parameters whose values are defined by experiments.

The proposed algorithm will generate a main topic for each annotation and a comparison with the ontology classes is performed in a descending order on the annotations list. If the topic is present in the ontology it is stored otherwise it is discarded. The list of candidate topics is then sent to the combination block.

The CNN classifier block is the core of image classification pipeline shown in Fig. 5.

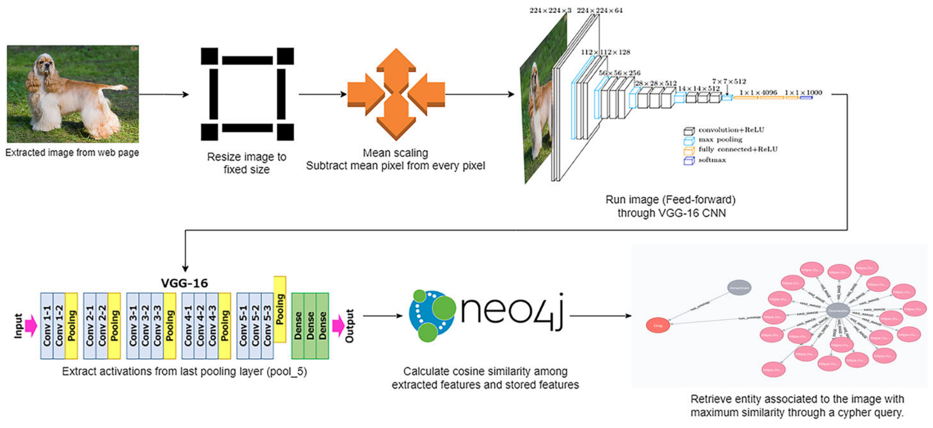
Extracted image from web page
Resize image to fixed size
Mean scaling
Subtract mean pixel from every pixel
224×224×3
224×224×64
112×112×128
56×56×256
28×28×512
14×14×512
7×7×512
1×1×4096
1×1×1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax
Run image (Feed-forward) through VGG-16 CNN
VGG-16
Input
Conv 1-1
Conv 1-2
Pooling
Conv 2-1
Conv 2-2
Pooling
Conv 3-1
Conv 3-2
Conv 3-3
Pooling
Conv 4-1
Conv 4-2
Conv 4-3
Pooling
Conv 5-1
Conv 5-2
Conv 5-3
Pooling
Dense
Dense
Dense
Output
Extract activations from last pooling layer (pool_5)
neo4j
Calculate cosine similarity among extracted features and stored features
Retrieve entity associated to the image with maximum similarity through a cypher query.

Figure 5 The proposed image pipeline analysis

In this article we use the VGG-16 architecture presented by a team of Google researchers in [54]. The authors argue that more hidden layers are added to a CNN a more accurate prediction will be obtained. Due to the outstanding results obtained by this CNN and the publicly available Keras model, VGG-16 CNN has been chosen for our purposes. This model was also adopted for content based image retrieval task in a similar fashion in [58, 60]

The images in our topic domain have been processed by the VGG-16 and the obtained vector features have been stored as a Neo4j node property.

The first step of the visual feature extraction process is to resize an image to a default size, in the case of the chosen CNN-model the size is 224px x 224px . A normalization step is also performed by subtracting from each pixel the mean of every other pixel [1]. Once we have a fixed size normalized picture, a feed-forward pass of the image is performed by the convolutional neural network.

Convolutional neural networks have the ability to learn features in a hierarchical fashion. Recent works have explored the possibility of extracting features from a given hidden layer of a CNN. In particular in [24] the authors argue that the features extracted from a CNN trained on a specific task can be used for other types of tasks that are unrelated to the original one. This is an important result for the purposes of this article because we will not further train the CNN having a general and task-independent system.

In the same article the authors also hypothesize that the neurons activations in its late hidden layers might serve as very strong features extractor for a variety of object recognition tasks. This statement has been confirmed by the authors' experiments. However, the authors in [8, 31, 59] highlight that features extracted from pooling layers can be used for image retrieval tasks which are the ones that we have carried out in this work. In particular, we extract features from the last max pooling layer of the CNN and use these features to perform a content based image retrieval.

Once the images are retrieved it's possible to infer their content and topic according to their semantics. The image semantics are well known thanks to the structure of the knowledge base. As a matter of fact, any image is linked to its topic and meaning in a Neo4j graphdb. In the same line of these considerations the features will be extracted from the pool-5 layer. A L2 normalization of the extracted feature vectors has been used because it improves accuracy following the results in the aforementioned works.

We use DeepLearning4J framework [20] to extract the visual features. The first step is to import the trained model, in our case the VGG-16 and after an initialization step it is ready to process images. The current analyzed image is first preprocessed as previously described then it is fed-forward to the convolutional neural network, where each layer will be responsible of a specific task such as convolution, pooling and non linearity.

As previously described, the layer activations are used as visual feature representation, and when an image is processed by the pool-5 layer its activations are extracted and stored.

The features of the analyzed web page images are extracted and compared with the ones stored in the Neo4j database.

The similarity measure used in our content based image retrieval process is the cosine similarity due to its performance in similar technology contexts and purposes [58].

Given a query image q and a generic image b:

$$S(q, b) = \frac{h(q) \cdot h(b)}{\|h(q)\| \|h(b)\|} \tag{4}$$

where q and b are the two images to compare and *h(q)* and *h(b)* the respective feature vectors.

In the case of a single image, we have to compare it to the rest of images in the feature database, and consider the maximum $b_{ret} = Max(S(q, b_k))$ between the query image q and all the other images b_k in our knowledge base.

b_{ret} is the the image associated with the maximum cosine similarity score w.r.t. the image q.

The topic represented by our image will be extracted from our database due to its knowledge structure.

The related information has been retrieved using Cypher queries. We can perform this task because in the Neo4j database structure there are specific relationships between images, documents and ontology concepts. Using these relationships an image depicting a mastiff (i.e. a specific dog bread) is related to the node *"Mastiff"*, which is in turn linked to the ontology node representing that concept. An example showing such structure is provided in Fig. 6 where we highlight also the expressive power of graph db visualization [16].

Once both the text and image classifications have been obtained, they have to be combined in order to have a final classification. The combination block will receive in input a list of candidate topics from both the image and text classifiers. The similarity score for an image is in a [0,1] range because a cosine similarity has been used. The same cannot be said for the text classification where the scores are simply the number of times an entity is found in the given page.

The following equation is used to have values between 0 and 1:

$$s_n(a_i) = \frac{e^{\beta \cdot s(a_i)} - e^{-\beta \cdot s(a_i)}}{e^{\beta \cdot s(a_i)} + e^{-\beta \cdot s(a_i)}} \tag{5}$$

where a_i is the i-th annotation, $s(a_i)$ is its score and β is an empirical parameter set by experiment [6, 35].

Now we are in the position of combining the scores from the two classifiers.

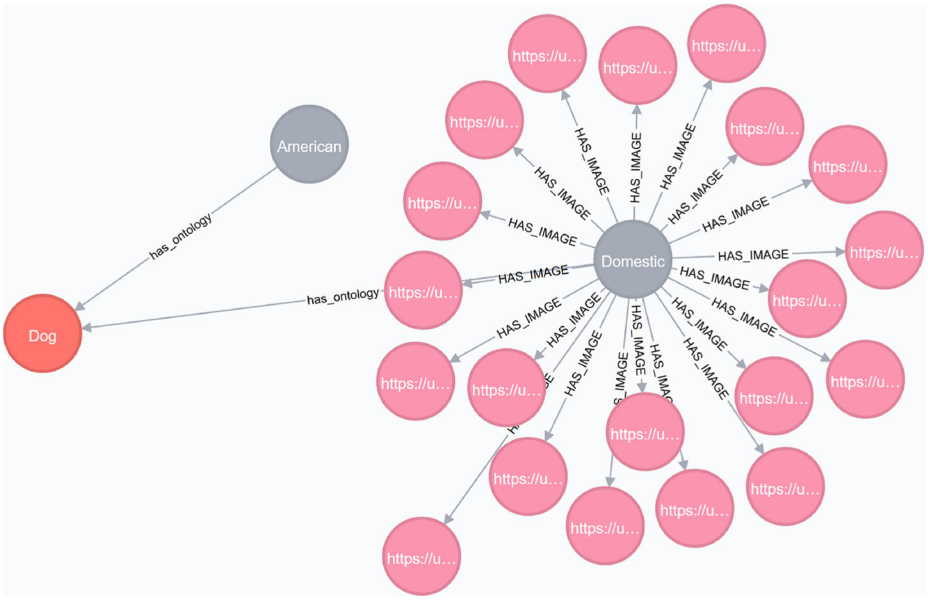

Figure 6 An example of the graph db structure

There are several approaches which combine classifiers and the SUM function gives very good performances [32, 46]. In this article a weighted sum will be used where the weights have been set by experiments under different contexts, in particular when there are no similar images in the Neo4j database. Another situation that have to be considered is when a near-perfect match is found. This situation happens when the almost-same picture is found. In this case it would be wrong to ignore it. Therefore, if the similarity $S(q, b_{ret}) \approx 1$, then the image classification is considered as a one take all strategy.

The combining classifier process is shown in Fig. 7.

When a given web page is downloaded and its topic is detected, we have to store it in our knowledge base. A new Neo4j node that represents the web page is created and linked to the found main-topic. A cypher query is used to handle the creation and linking to the crawled web-page. A relation of "HAS_RESOURCE" exists between the main topic entity and the classified web page. All the images that have been found in the crawled web page are also linked to the created node. The overloading in the creation of too many image nodes is avoided by means of a parameter in the crawler configuration file which sets a maximum number of images per web site to be stored.

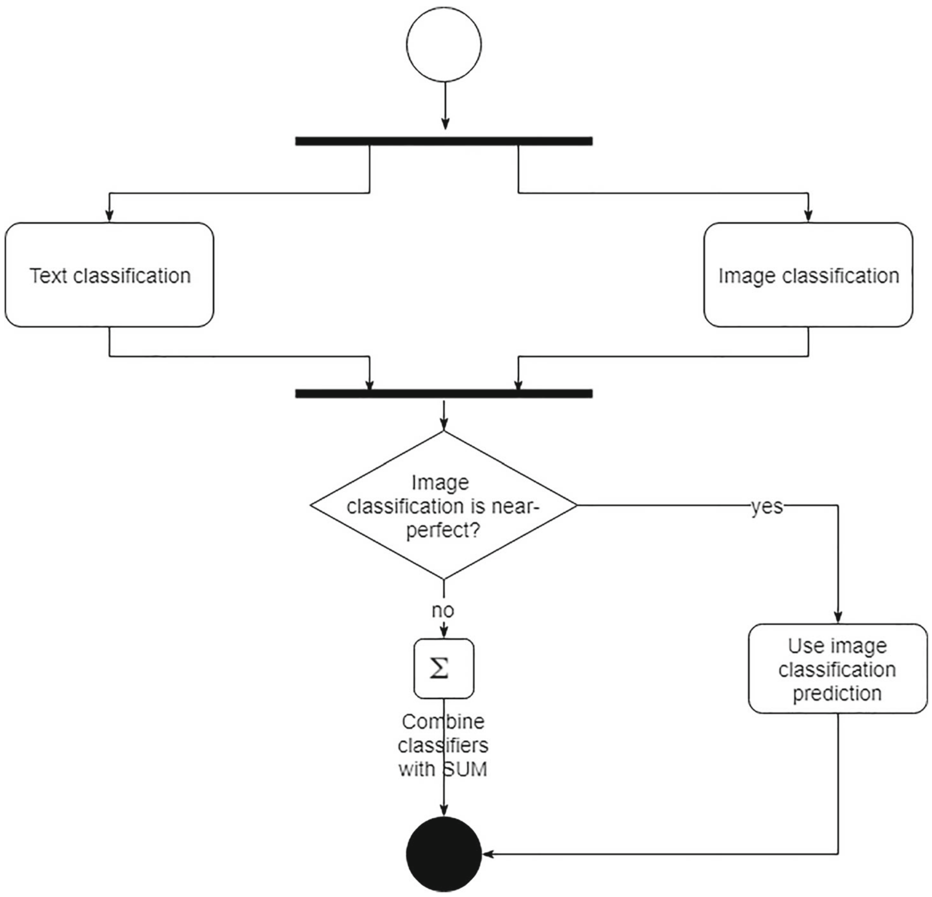
Text classification
Image classification
Image classification is near-perfect?
yes
no
Σ
Combine classifiers with SUM
Use image classification prediction

Figure 7 Combining textual and image classifications

4 Experimental results

The performances evaluation of a focused crawler is a hard task and it presents several challenges. A first challenge is to define which kind of measures should be used in order to evaluate the accuracy of the crawling process.

A first approach could be in the use of a categorized web pages dataset comparing the human classification with the ones provided by the crawler. However, this approach requires to manually label a sub-graph of the web, and since the crawler needs to work with large amounts of web pages the labeling effort would be too high. Another metric that could be used is the one proposed in [18], the authors state that a focused crawler should be evaluated on the base of the harvest ratio which is defined as the rate of relevant page acquisition and how irrelevant pages are discarded off a crawling session. We can formalize the harvest ratio as:

$$Hr = \frac{Relevant\_pages}{Crawled\_pages} \tag{6}$$

A larger harvest ratio implies better performances because it means that the crawler discovers a web subgraph full of relevant pages.

To assess the relevance of a web page assigned by an automatic web crawler, its outputs must be compared with a ground truth. The ground truth is determined by humans. In the web information retrieval task, where the system may be evaluated using corpora consisting of tens of thousands of documents, it would be almost impossible to judge the relevance of all documents with respect to all queries used in the evaluation. Moreover, in a real scenario as the web this is literally impossible due to the size of the web itself.

We develop an automatic process to recognize if a given page is relevant for the domain of out focused crawler using a an experimental setup similar to [55].

A ground truth related to the crawler topic (i.e. animal domain) will be created, then a measure of similarity will be computed between the crawled page and each document in the ground truth. If the arithmetic mean of the similarity scores is greater than a predefined value, the page will be judged to be relevant.

Given a crawled page C_i and a set of relevant ground truth documents R, a similarity score $sim(C_i, R_i)$ can be computed between C_i and a page R_i extracted from R. We define the average similarity as the arithmetic mean of the similarity scores:

$$AS(C_i) = \frac{1}{|R|} \sum_{R_i \in R} sim(C_i, R_i) \tag{7}$$

A web page is defined to be relevant if $AS \geq threshold$. The threshold value has been set to 0.7 following the results in [55] Once the ground truth is created and a seed URL is chosen, different type of crawlers can be initialized with the same seed URL. For each crawled page, using the aforementioned strategy, an average similarity is computed and a crawled page is set to be relevant or irrelevant.

Given N crawled pages, the harvest ratio is:

$$HR = \frac{|V|}{|N|} \tag{8}$$

where $|V|$ is the cardinality of the set of web pages that satisfy the $V_j \in V \rightarrow AS(V_j) \geq th$ condition.

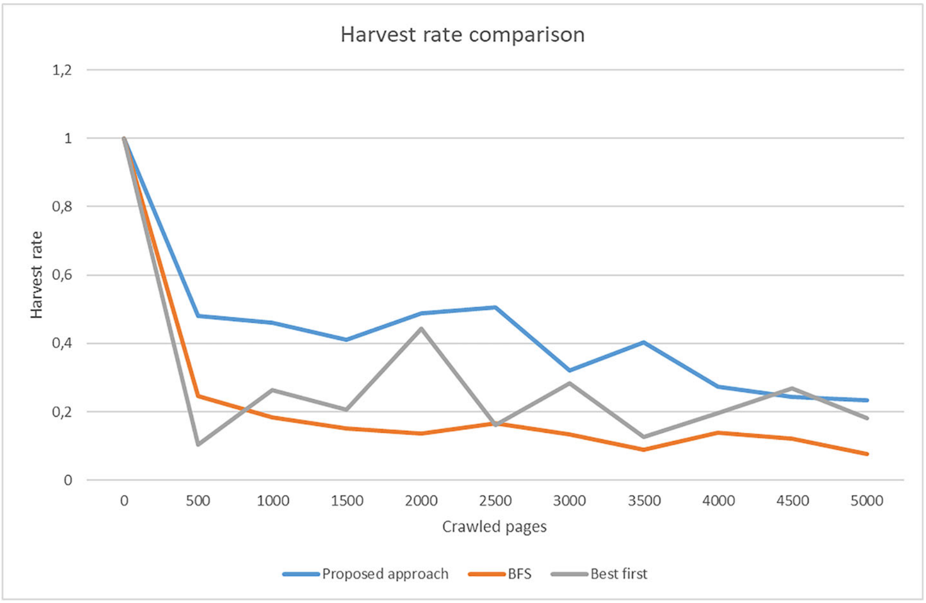

Figure 8 Harvest rate comparison

We compare our algorithm with the breadth first and the best first approaches. The experimental results are shown in Fig. 8. It is possible to argue how the proposed approach improves the average of the harvest rate of the given pages. As a matter of fact the average harvest ratio of the proposed crawler is higher then the one of the other two approaches.

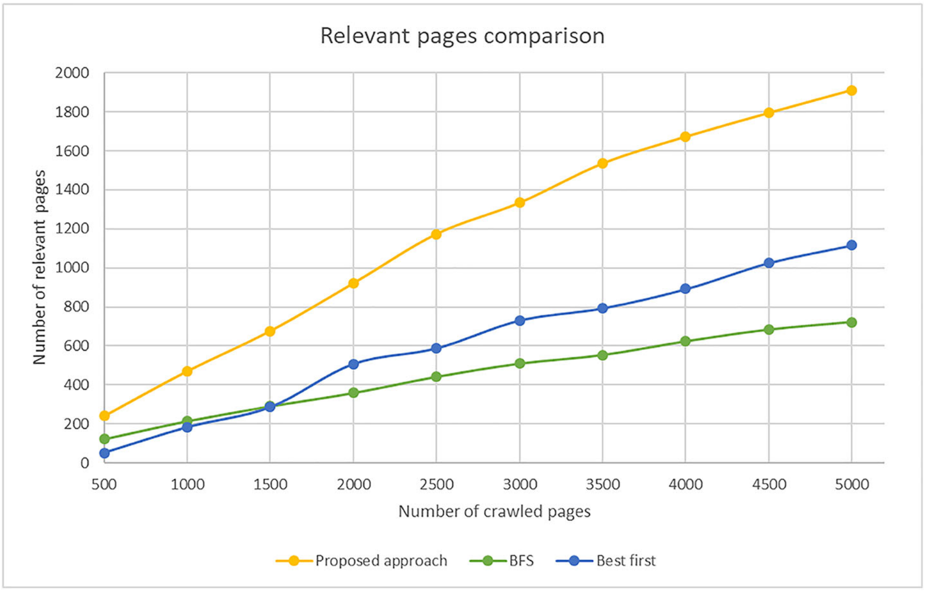

Figure 9 Harvest rate comparison

The obtained improvement is also show in Fig. 9 where the comparison between the three algorithms is visualized in terms of relevant pages span over time. In this figure the difference of performances among the analyzed approaches is further highlighted.

5 Conclusion and future work

In this article an ontology-driven focused crawler based on the combination of image and text classification techniques has been presented. The system integrates modern technologies and novel algorithms achieving good results with regards to a well know measure as harvest ratio compared with state of the art techniques.

We highlight how heterogeneous technologies such as convolutional neural network and linked open data can co-exist to achieve significant improvements in hard information retrieval tasks, and how different classifiers can have benefit from each other to improve the overall result.

The built system makes extensive use of deep learning and takes advantage of such powerful technologies, confirming that convolutional neural networks are an important tool to improve several image analysis tasks such as retrieval, classification and detection. The use of features extracted from CNN is also explored and they are used to enhance the crawling process. Linked open data have an important role in information retrieval tasks and this work follows the general consensus on the use of the semantic web technologies in complex application context as the web. Moreover, the use of formal structures to represent knowledge as ontologies allow us to have a high level of generalization in different conceptual domains.

The implementation of graph database as knowledge repository has a great impact on the efficient of the whole framework and it greatly improve the overall crawling experience providing a native graph structure where properties among concepts, entities and documents are clearly represented.

The proposed architecture is highly modular and reusable, making the process of integrating other tools very simple. Several new research line could be further investigated as the merge of domain specific ontologies on our knowledge base. In this context schema matching techniques and automatic population of very large knowledge bases implemented as a graph db [15, 17] together with the integration of multimedia components [50] are an interesting research domain

Moreover, the accuracy of our crawler could be improved by training the convolutional neural network on specific domains used in the focused crawler. It would also be enhanced by the classification task to recognize the different objects within a given image by using a semantic instance segmentation approach. Using this technique, it would be possible to have multiple entities classification on a single image, whereas with the one-image one-classification approach we are limited to reduce images with several complex objects to an unique categorization.

References

1. Building an image classification web application using vgg-16 - deeplearning4j: Open-source, distributed deep learning for the jvm. https://deeplearning4j.org/build_vgg_webapp. (Accessed on 03/26/2018)

2. Abualigah L, Qasim M, Hanandeh ES (2015) Applying genetic algorithms to information retrieval using vector space model. Int J Computer Sci Eng Appl 5(1):19
3. Abualigah LM, Khader AT (2017) Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. J Supercomputing 73(11):4773–4795
4. Abualigah LM, Khader AT, Hanandeh ES (2018) Hybrid clustering analysis using improved krill herd algorithm. Appl Intell 48(11):4047–4071
5. Aggarwal CC, Zhai CX (2012) A survey of text classification algorithms. In: Mining text data. Springer, pp 163–222
6. Albanese M, Capasso P, Picariello A, Rinaldi AM (2005) Information retrieval from the web: an interactive paradigm. In: International workshop on multimedia information systems. Springer, pp 17–32
7. Allahyari M, Pouriyeh S, Assefi M, Safaei S, Trippe ED, Gutierrez JB, Kochut K (2017) A brief survey of text mining: classification, clustering and extraction techniques. arXiv:1707.02919
8. Babenko A, Lempitsky V (2015) Aggregating local deep features for image retrieval. In: Proceedings of the IEEE international conference on computer vision, pp 1269–1277
9. Baeza-Yates R, Ribeiro-Neto B et al (1999) Modern information retrieval, vol 463. ACM Press, New York
10. Batsakis S, Petrakis EGM, Milios E (2009) Improving the performance of focused web crawlers. Data Knowledge Eng 68(10):1001–1013
11. Bergman MK (2001) White paper: the deep web: surfacing hidden value. J Electron Publishing, 7(1)
12. Caldarola EG, Picariello A, Rinaldi AM (2015) Big graph-based data visualization experiences: the wordnet case study. In: IC3K 2015 - Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, vol 1, pp 104–115
13. Caldarola EG, Picariello A, Rinaldi AM (2016) Experiences in wordnet visualization with labeled graph databases. Commun Comput Inform Sci 631:80–99
14. Caldarola EG, Rinaldi AM (2015) Big data: a survey: the new paradigms, methodologies and tools. In: DATA 2015 - 4Th international conference on data management technologies and applications, proceedings, pp 362–370
15. Caldarola EG, Rinaldi AM (2016) An approach to ontology integration for ontology reuse. In: Proceedings - 2016 IEEE 17th International Conference on Information Reuse and Integration, IRI 2016, pp 384–393
16. Caldarola EG, Rinaldi AM (2017) Big data visualization tools: a survey: the new paradigms, methodologies and tools for large data sets visualization. In: DATA 2017 - Proceedings of the 6th International Conference on Data Science, Technology and Applications, pp 296–305
17. Caldarola EG, Rinaldi AM (2018) A multi-strategy approach for ontology reuse through matching and integration techniques. Advan Intell Syst Comput 561:63–90
18. Chakrabarti S, Van den Berg M, Dom B (1999) Focused crawling: a new approach to topic-specific web resource discovery. Comput Netw 31(11-16):1623–1640
19. Cho J, Garcia-Molina H, Page L (1998) Efficient crawling through url ordering. Comput Netw ISDN Syst 30(1-7):161–172
20. Chris Nicholson A, Gibson A (2017) Deeplearning4j: Open-source, distributed deep learning for the jvm. Deeplearning4j org
21. Maurice de Kunder (2016) The size of the world wide web (the internet). Hentet 15
22. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: CVPR09
23. Diligenti M, Coetzee F, Lawrence S, Giles CL, Gori M et al (2000) Focused crawling using context graphs. In: VLDB, pp 527–534
24. Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2014) Decaf: a deep convolutional activation feature for generic visual recognition. In: International conference on machine learning, pp 647–655
25. Ehrig M, Maedche A (2003) Ontology-focused crawling of web documents. In: Proceedings of the 2003 ACM symposium on Applied computing. ACM, pp 1174–1178
26. Glover EJ, Tsioutsiouliklis K, Lawrence S, Pennock DM, Flake GW (2002) Using web structure for classifying and describing web pages. In: Proceedings of the 11th international conference on World Wide Web. ACM, pp 562–569
27. Gong Y, Wang L, Guo R, Lazebnik S (2014) Multi-scale orderless pooling of deep convolutional activation features. In: European conference on computer vision. Springer, pp 392–407
28. Gruber TR (1993) A translation approach to portable ontology specifications. Knowledge Acquisition 5(2):199–220

29. Hassan T, Cruz C, Bertaux A (2017) Ontology-based approach for unsupervised and adaptive focused crawling. In: Proceedings of The International Workshop on Semantic Big Data. ACM, p 2
30. Ji W, Wang D, Hoi SCH, Pengcheng W, Zhu J, Zhang Y, Li J (2014) Deep learning for content-based image retrieval: a comprehensive study. In: Proceedings of the 22nd ACM international conference on Multimedia. ACM, pp 157–166
31. Joe Y-HN, Fan Y, Davis LS (2015) Exploiting local features from deep networks for image retrieval. arXiv:1504.05133
32. Kittler J, Hatef M, Duin RPW, Matas J (1998) On combining classifiers. IEEE Trans Pattern Anal Mach Intell 20(3):226–239
33. Kosala R, Blockeel H (2000) Web mining research: a survey. ACM Sigkdd Explorations Newsletter 2(1):1–15
34. Lefteris K (2008) An ontology-based focused crawler. In: International conference on application of natural language to information systems. Springer, pp 376–379
35. Li Y, Bandar ZA, McLean D (2003) An approach for measuring semantic similarity between words using multiple information sources. IEEE Trans Knowledge data Eng 15(4):871–882
36. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26
37. Mendes PN, Jakob M, García-Silva A, Bizer C (2011) Dbpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th international conference on semantic systems. ACM, pp 1–8
38. Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38(11):39–41
39. Mohammad L, Abualigah Q (2019) Feature selection and enhanced krill herd algorithm for text document clustering. Springer, Berlin
40. Mukhopadhyay D, Biswas A, Sinha S (2007) A new approach to design domain specific ontology based web crawler. In: 10th International Conference on Information Technology (ICIT 2007). IEEE, pp 289–291
41. Najork M, Wiener JL (2001) Breadth-first crawling yields high-quality pages. In: Proceedings of the 10th international conference on World Wide Web. ACM, pp 114–118
42. Novak B (2004) A survey of focused web crawling algorithms. Proc SIKDD 5558:55–58
43. Oquab M, Bottou L, Laptev I, Sivic J (2014) Learning and transferring mid-level image representations using convolutional neural networks. In: 2014 IEEE conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp 1717–1724
44. Pant G, Srinivasan P (2005) Learning to crawl: comparing classification schemes. ACM Transactions on Information Systems (TOIS) 23(4):430–462
45. Picariello A, Rinaldi AM (2007) Crawling the web with ontodir. In: International conference on database and expert systems applications. Springer, pp 730–739
46. Purificato E, Rinaldi AM (2018) Multimedia and geographic data integration for cultural heritage information retrieval. Multimed Tool Appl 77(20):27447–27469
47. Qi X, Davison BD (2009) Web page classification: features and algorithms. ACM Computing Surveys (CSUR) 41(2):12
48. Razavian AS, Azizpour H, Sullivan J, Carlsson S (2014) Cnn features off-the-shelf: an astounding baseline for recognition. In: 2014 IEEE conference on Computer vision and pattern recognition workshops (CVPRW). IEEE, pp 512–519
49. Razavian AS, Sullivan J, Carlsson S, Maki A (2016) Visual instance retrieval with deep convolutional networks. ITE Trans Media Technol Appl 4(3):251–258
50. Rinaldi AM, Russo C (2018) A matching framework for multimedia data integration using semantics and ontologies. In: 2018 IEEE 12Th international conference on semantic computing (ICSC). IEEE, pp 363–368
51. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
52. Sebastiani F (2002) Machine learning in automated text categorization. ACM Computing Surveys (CSUR) 34(1):1–47
53. Sharma DK, Khan MA (2015) Safsb: a self-adaptive focused crawler. In: 2015 1st International Conference on Next Generation Computing Technologies (NGCT). IEEE, pp 719–724
54. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
55. Yajun D, Liu W, Lv X, Peng G (2015) An improved focused crawler based on semantic similarity vector space model. Appl Soft Comput 36:392–407
56. Yang S-Y (2010) Ontocrawler: a focused crawler with ontology-supported website models for information agents. Expert Syst Appl 37(7):5381–5389

57. Yohanes BW, Handoko H, Wardana HK (2013) Focused crawler optimization using genetic algorithm. TELKOMNIKA (Telecommunication Computing Electronics and Control) 9(3):403–410
58. Zhang F, Zhong B-J (2016) Image retrieval based on fused cnn features DEStech Transactions on Computer Science and Engineering (aics)
59. Zhi T, Duan L-Y, Wang Y, Huang T (2016) Two-stage pooling of deep convolutional features for image retrieval. In: 2016 IEEE International Conference on Image processing (ICIP). IEEE, pp 2465–2469
60. Zhou Zz, Zhang L (2017) Content-based image retrieval using iterative search. Neural Process Lett 1–13

Andrea Capuano has graduated cum laude in computer engineering at University of Napoli Federico II. His research and application interests include artificial intelligence, information retrieval and data mining with a particular focus on scalable software solutions aided by Artificial intelligence.

Antonio M. Rinaldi has graduated in Computer Engineering at the University of Napoli Federico II and he received his PhD in Computer Science and Systems in 2005 from the same university, with a dissertation on the use of ontologies and semantic relatedness for information retrieval on the web. In the last years, he has been involved in several research and industrial projects related to information retrieval, multimedia database integration and geographic information systems. His fields of interest are: knowledge representation and retrieval, Big Data, multimedia database and GIS. Currently, he is an Assistant Professor at DIETI Dipartimento di Ingegneria Elettrica e delle Tecnologie dellInformazione and he is the Director of the IKNOS-LAB Intelligent and Knowledge Systems, LUPT, University of Napoli Federico II.

Cristiano Russo received the MSc degree from the University of Naples, Federico II, Napoli, Italy, in 2017. He is currently a PhD student in joint supervision between University Federico II and Universit Paris- Est. His current research interests include information retrieval, ontologies, machine learning and data mining.