I.J. Information Technology and Computer Science, **2014, 09, 52-58**
Published Online August 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.09.07

An Approach for Indexing Web Data Sources

Saidi Imene
LITIO laboratory, University of Oran, BP 1524, El-M'Naouer, 31000, Oran, Algeria
Email: saidi.imene@univ-oran.dz

Nait Bahloul Safia
LITIO laboratory, University of Oran, BP 1524, El-M'Naouer, 31000, Oran, Algeria
Email: nait-bahloul.safia@univ-oran.dz

Abstract— Web information sources such as forums, blogs, and news articles are becoming increasingly large and diverse. Even if advances in technology are helping to improve techniques for dealing with the large amounts of the generated data, such data sources are heterogeneous in structure (semi structured or unstructured sources) and nature (texts or images). Implementation of software solutions is then necessary to prepare data and access these sources in a homogenous way. In this paper we present an approach for indexing heterogeneous data sources. Our objective is to offer techniques for efficient indexing of web sources by storing only the necessary information. We propose automatic indexing for semi structured or unstructured sources (e.g., xml files, html files) and annotation for other sources (e.g., images, videos that exist within a page). We present our algorithms of indexing and propose the use of MapReduce model to build a scalable inverted index. Experiments on a real-world corpus show that our approach achieves a good performance.

Index Terms— Information Retrieval, Indexing Techniques, Data Mining, Mapreduce

I. Introduction

One of the main difficulties encountered by the users of the web is heterogeneity of information sources and their diversity. Heterogeneity of sources may come from the format or the structure (structured sources: relational databases, semi structured sources: XML documents, or unstructured: texts). One of the objectives of the future web, which is called *semantic web,* is to provide mechanisms to access heterogeneous data sources in a standardized way. The systems known as mediation systems are then very useful, in the presence of heterogeneous data, because they give the impression of using homogeneous system.

The main objective of this paper is to prepare data to be used in a global architecture for querying heterogeneous data sources of the web. We then index data sources syntactically and semantically and enable the exploitation of the generated indexes in order to use them in a global system.

This paper is organized as follows: In the next section, we present some related works to our approach. Section 3, is reserved for the description of our approach in order to show the different steps for the creation of indexes. In section 4, we present our algorithms and in the section 5, we present experiments. A conclusion is presented to synthesize this work and a set of prospects are proposed in the end of this paper.

II. Related Works

The scientific community has long been interested in indexing as a critical step of pretreatment of data and an important process in information retrieval systems. The implementation of software solutions which improve performance of the systems has always been necessary.

The new models such as MapReduce handle quantities of data that are becoming increasingly large. To improve execution time and data processing, several implementations have been proposed, among these solutions, we can find *Hadoop* which is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment [1]. Some research has been implemented to evaluate performance of the MapReduce model [2] [3].

Several studies have been published about indexing strategies and MapReduce jobs. A good indexing strategy will allow fast access to desired data and a short time for the indexing process. Among the existing works on the problem of indexing, some approaches [4] [5] [6] analyze the query workload and decide which attributes to index. Another research direction [7] is the use of distributed key-value storage system as a generic infrastructure to accommodate structured and semi-structured data.

Lin et al. [8] propose to build a full-text inverted index at the level of the blocks to optimize regular-expression queries over text fields.

In our approach, we propose to index the concepts extracted from a semantic indexing phase after a filtering phase (syntactic phase). The creation of inverted index by storing only the concepts and the use of MapReduce model makes the index scalable.

III. Our Approach

In this section, we present our approach of indexing heterogeneous data sources, i.e the steps of the construction of indexes, including how we adapt the classical techniques of indexing.

First of all, we present our approach and remind some notions of indexing.

Depending on the type of the source, two treatments are possible (Fig. 1):

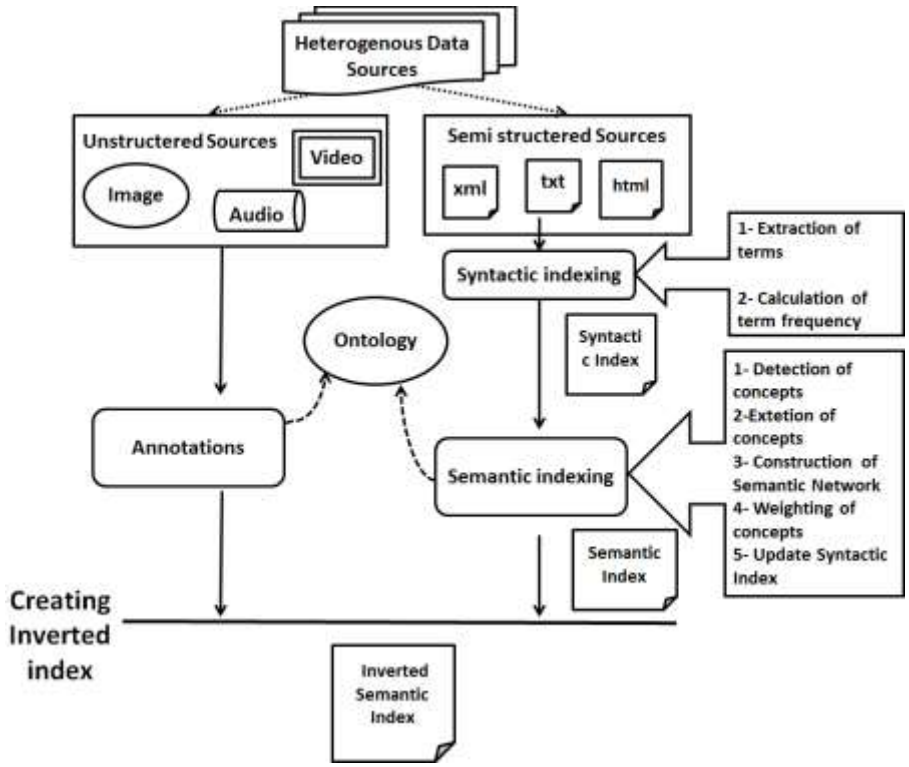

Fig. 1. Overall Scheme of our Approach

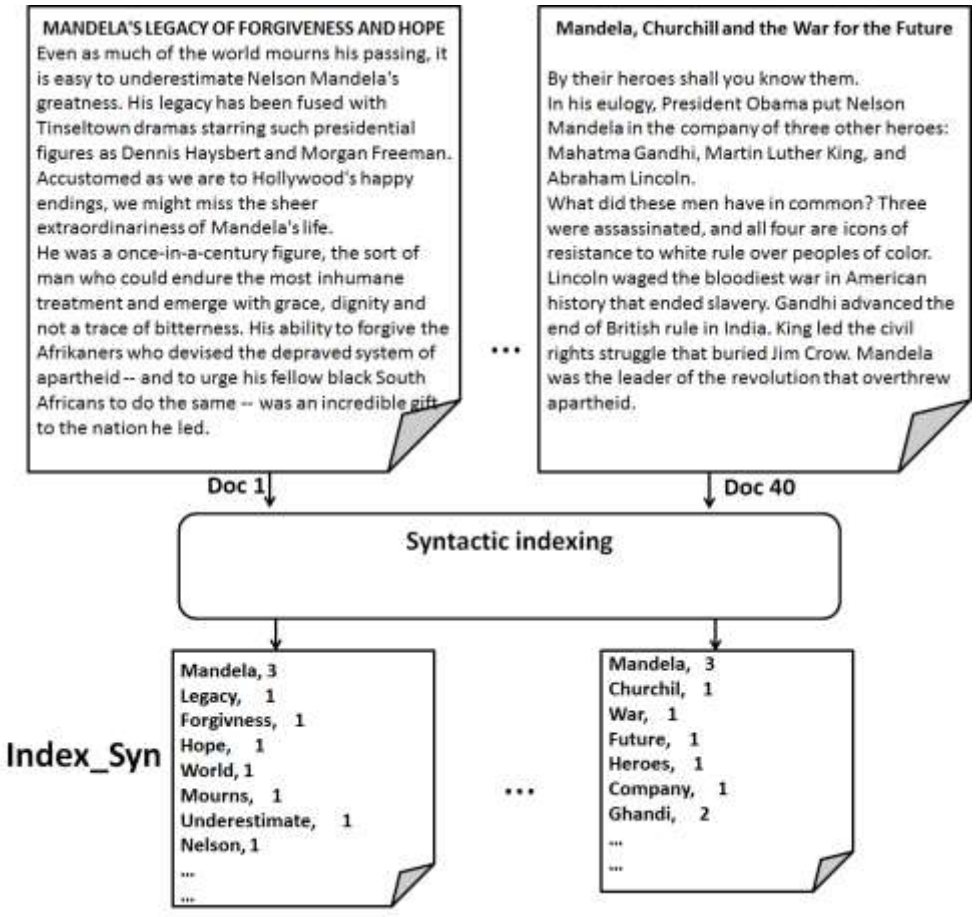

Fig. 2. Syntactic indexing

- ***Automatic indexing:*** concerns the semi structured or unstructured sources, i.e: textual documents, ex: text files.
- ***Annotation:*** (tagging) concerns annotations and using extracted concepts for the creation of inverted index. Annotation is for non-textual sources, e.g. video, image.

A. Automatic indexing

In this type of indexation, we had been inspired by the work of Baziz et al [9] [10] [11].

The construction of indexes in the automatic indexing is done in two main phases: the syntactic indexation and semantic indexation (Fig. 1).

1. Syntactic Indexation

The objective of the syntactic indexation is to extract the terms from the documents and store them with their number of occurrences in a physical index. This type of indexing has been presented in several works [12] [13].

a- Extraction of words

In this first phase of construction of indexes, we extract the significant terms of the document (not a stop word).

b- Calculation of the occurrence of the word:

For the significant words, the frequency of occurrences must be calculated. Every time a word is encountered in the document, we increment its frequency (Term Frequency).

The result of this phase is the file "Index_Syn": Syntactic Index related to each document. It is constituted as follows:

For each document, you can find all the words which belong to the document with their term frequencies.

Document_ID => {(term$_1$, tf_1)… ($Term_n$, tf_n)} / n: Number of Significant words of the document.

An example of this type of indexing is given in the following Figure (Fig. 2).

Each document of the corpus is treated and significant terms are extracted with their TF (Fig. 2).

2. Semantic Indexing

This phase accepts as input information from the syntactic indexation.

Semantic indexing consists of the following steps. In *Index_Syn* (created from the syntactic indexation) and for each document, we do:

a- Detection of concepts

A term is said concept if it belongs to at least one entry in the ontology. WordNet [21] is used to do the matching.

The detection of concepts is done by taking the words one by one and projecting them on the ontology to detect the concepts. Concepts are identified from WordNet and marked in a document.

For example: "president" is a concept because it matches an entry in the ontology which is of the type "person". This step is the projection of the document on the ontology.

b- Expansion of concepts using the ontology

For each detected concept, a specific treatment is done. It is to detect links between the different concepts and to link them together using WordNet.

Concepts are then extended by their synonyms and hyponyms that exist in the document.

A list of synonyms and meaning will be awarded to each concept. Creation of links and relations between concepts forms some kind of a network for each concept. This step is the projection of the ontology on the document.

c- Construction of the semantic network

In the previous two steps, we used ontology to represent the content of the documents in the form of:

- *Concepts.*
- *Relations between concepts.*

We had as results and for each concept a number of terms which are associated with. We call this combination the *semantic network of the term*.

Definition:

The set of terms $Sn = \{t_1 \ldots t_p\}$ forms a network with a term *t* of a document doc, means that: - *Sn* is formed by the union of the elements e_i / e_i are terms in relation with the term *t*, where the relation is = {synonymy, derivation, same familly, ... }.- AND $\forall\, t\,(e_i) \in Sn$, $t\,(e_i) \in doc$.

d- Weighting of concepts

There are several approaches [19] [20] to weight the significant terms of a document or a query. Many of them are based on the factors T*f* and *idf* which allow the consideration of the local and global weights of a term.

The measure *tf*idf* allows to approximate the representativeness of a term in a document.

In our approach, we will use a variant of this measure, which is the *Cf*idf*.

- *Calculation of CF*

The CF is not the frequency of the term but that of the concept. To be calculated, we must add the *tf* of the initial term to all *tf* of the terms that have been associated in the same document in the previous phase, i.e the *tf* of the terms of the semantic network, as shown in (1).

$$\mathrm{Cf} = \mathrm{tf}\,(\mathrm{t}) + \sum \mathrm{tf}\,(\text{semantic network}). \quad (1)$$

Such as *t* is the current term and *tf (*semantic network) are all terms of the semantic network.

- *Calculation of idf*

In this work, we distinguished the frequency of occurrence of a term in a document (term frequency, *tf*) which was already calculated in the Syntactic index "*index_syn*" and the frequency of occurrence for the same term in the entire collection (inverse document frequency, *idf*) , as shown in (2).

$$\mathrm{Idf} = (\log((\text{total documents}) / (\text{number of documents with the term})). \quad (2)$$

Idf = (log ((total documents) / (number of documents with the term))[1].

e- Updating of index_Syn

Index_Syn is updated by taking into account the concepts with an entry in the ontology (the detected concepts).

The terms which belong to their network will be taken as the concepts with the same weight (which is the sum)

In this phase, we enrich the index_Syn with the terms of the created network and we remove the terms that are not concepts.

The result will be, a semantic index (Index_Sem) which is composed as follows:

{{(concept_1_k;i), (concept1_{k+1};i)}, …,
{(concept_n_k;j), (concept_n_{k+1};j), …}}

i, j: Weighting of the concept (Cf.idf already calculated). k: 1...m / m: the size of semantic network of the concept.

B. Annotation

For non-textual data sources, we propose the use of semantic annotation tools. Several works present tools for image annotation [15] [16] and video annotation [17] [18]. This part of the work will be presented in a future work.

Concepts are related to sources and are then used to create the inverted index.

- **Creation of the inverted index**

This step is necessary to allow exploitation of the indexes.

The creation of inverted index is to correspond to all concepts, the documents that contain them with their weights.

Concept 1 --> <Documents and weighting>
…
Concept n --> <Documents and weighting>

IV. Algorithms

In this section, we propose efficient algorithms of the different phases of our approach. We also introduce the use of MapReduce model to create inverted index.

Syntactic indexing phase (removes stop words and calculates Tf) is summarized by the following algorithm (Algorithm 1):

The previous algorithm (Algorithm 1) extracts the terms, calculate the *Tf* (term frequency) of each term and generate a syntactic index of the documents.

Semantic indexing phase is summarized by the following algorithm (Algorithm 2):

Algorithm 1: Syntactic indexing

```
Require: documents: corpus
Output: Index_Syn: syntactic index

For each Doc_in (documents) do
/* for all documents of the corpus, current is Doc */;
    Create (Doc.index_Syn);
    /* create index_Syn of each document */;
        For each line_in (Doc) faire
            /*for all the lines of document, current is
            line*/;
            For each   word_in (line);
            /*for all the words of the line*/;
     If stop_word (word) = False then
     /* If the word is not a stop word */;
        If Doc.Index_Syn.Contains (word) = False then
        /* Index_Syn does not contain the word */;
        Doc.Index_Syn.Add (word, 1);
       /*add the word to index, tf =1 term frequency*/;
        ELSE
       /*Index_Syn contain the word: increment tf*/;
        Doc.Index_Syn.IncrementTf (word.TF);
     END If;
     END If;
         END For;
      ENDFor;
ENDFor;
END
```

Algorithm 2: Semantic Indexing

```
Require: Index_Syn: index
Output: Index Sem: index semantic

For each index_Syn do
/*for all indexes Index_Syn of documents*/;
   For each term in index_Syn do
   /*for all terms of Index_Syn*/;
            If term.isConcept () then
            /*Term corresponds to an input of
            ontology*/;
            /*Projection on ontology*/
            Expansion _with_ontology ();
            /* detect links between concepts and extend
            them*/;
            Construction_semantic_network ();
            /*concept and the terms associated with*/;
            Ponderation_Concept ();
         /*concept is weighted according to Cf.idf */;
            Else /*Term is ignored*/;
            END If;
Update_Index_Syn ();
/*update Index_Syn -> only concepts are considered*/;
    END For;
END For;
END
```

The previous algorithm (Algorithm 2) extracts the concepts that will be considered in the semantic index (by projecting syntactic indexes of the documents on an ontology of the considered domain), and extends them by creating the semantic network and weighting the concepts. This algorithm generates semantic indexes of the documents.

[1] http://en.wikipedia.org/wiki/Tf-idf

The following algorithm (Algorithm 3) creates an inverted index from semantic indexes of the documents:

Algorithm 3: Creation of inverted index
Require: Index_Sem: semantic indexes
Output: Index_Sem_Inv: Inverted semantic index

```
For each Index_Sem do
/*for all indexes Index_Sem of documents*/;
  For each Concept in index_Sem do
  /* for all concepts of all the semantic indexes of the
   documents*/;
     If Concept.existsIn (Index_Sem Inv ()) = True
     Then
       /*If the Concept exists in Index_Sem_Inv */;
           Index_Sem_Inv [Concept].Add [Index_Sem.Do
           cument, weighting];
           /*add document of current Index_Sem to
           Index_Sem_Inv with the weighting of the
           concept*/
     Else /*Concept does not exist in the inverted index*/;
     Index_Sem_Inv [end].Add [Concept,
     Index_Sem.Document, weighting];
           /*add the concept with the document and
           weighting at the end of the inverted index*/;
     END If;
  END For;
END For;
END
```

The previous algorithm (Algorithm 3) creates from all the semantic indexes of the documents, an inverted index (Index_Sem_Inv) that matches all the concepts with their documents and weightings.

- **MapReduce Algorithm**

Map-Reduce framework [22] is a popular way to deal with a large scale of data. In this paper, we focus on indexing which is a crucial step made offline. The use MapReduce framework grantee the performance over big data sets.

To create an inverted index, we use the pseudo code given in [14] and we adapt it to our approach.

The creation of the inverted index (Index_Sem_Inv) of our approach is as follows:

```
1: Class Mapper
2: procedure Map (docid id, doc d)
3: H <– new AssociativeArray;
4: for all Concept C in Index_Sem do
5: cf{C} <– cf{C} + 1;
6: for all Concept C in H do
7: Emit (Concept C, posting <id, cf{C}>)

1: Class Reducer
2: procedure Reduce (concept c, postings [< id1, cf1> ...])
3: P <– new List;
4: for all posting <a, f> in postings [<id1, cf1> ...] do
5: Append (P, <a, f>);
6: Sort (P)
7: Emit (Concept C, postings P)
```

The Map function calculates all the *cf* (Concept frequency) and emits the concept with a Posting List containing the identifier of the document and cf of the concept in this document. The Reducer groups and ranks the positing lists of each concept, i.e, all documents that contain the concept and the corresponding cf. Reducer emits the concept and its Postindg list.

V. Experiments

In order to analyze our approach and study its performance, we considered the execution time and the size of the created indexes. We will present in this section some results and compare our approach of indexing using MapReduce to a classical one that uses Algorithm 3 of the previous section (Section 4) without MapReduce i.e without parallelization of execution; we name it "Approach 2".

We have created a corpus consisting of 20000 image-text pairs, retrieved from the Yahoo! News website[2].

Experiment 1: Average execution time

In this series of experiments, we measured the average execution time of the creation of indexes. We have launched the two approaches (with and without MapReduce) and calculate the time.

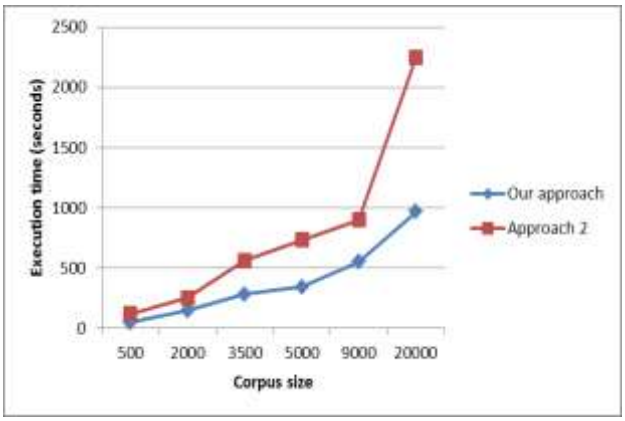

Fig. 3. Execution time by varying the size of corpus

Fig. 3 shows the results of execution time of the two approaches by varying the size of corpus (we used various sizes of corpus from 500 files per corpus to 20000 files). The execution time (time for the creation of index) given by our approach is 282 seconds for example when the corpus size is 3500 when the Approach 2 is 564 seconds. In all cases, the required time for the creation of index of our approach is better than approach 2 (see Fig. 3). We can deduce from these results that our approach gives better results, which is estimated 49 % faster (the average considering the results of Fig. 3).

Experiment 2: Index size

This second experiment is to study size of created index by varying the corpus size.

Fig. 4 summarizes the results obtained by this experiment. We vary the size of corpus from 500 files to 20000 files, as shown in Fig. 4.

[2] http://news.yahoo.com

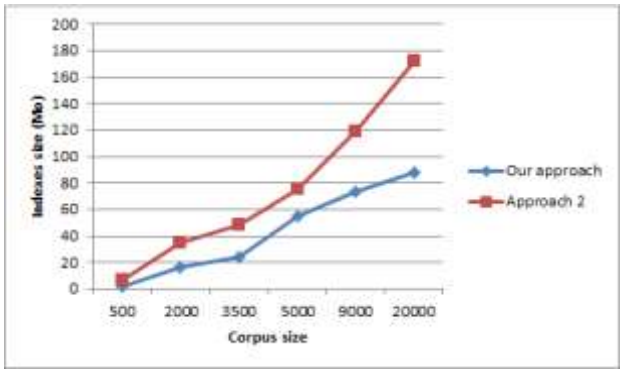

Fig. 4. Index size by varying the size of corpus

We note that our approach reduces the size of the created index. For the examples shown in Fig. 4, the size of index is 24Mo for our approach when the corpus size is 3500 and the index size of approach 2 is 48.5Mo when the corpus size is 3500, we have an average gain of 48% of space in our approach (the average considering the results of Fig. 4), we deduce that the proposed approach reduce the necessary space for index because of the performance of the MapReduce model and the storage of the necessary information (concepts only).

VI. Conclusion

Indexing plays an important role in information retrieval. There are several types of indexing but in all cases the principle is to convert data sources to computerized data. The objective of this article is to propose techniques for automatic semantic indexing (indexing linked to ontology) on heterogeneous sources.

In this paper, we proposed an approach for indexing semi or unstructured sources. Annotation is proposed for non-textual sources and automatic indexing for textual sources by considering syntactic and semantic indexing.

The goal of the integration of techniques for diverse sources is to manage the heterogeneity, and consider all sources of information of the web. To create our inverted index, we used MapReduce Framework to allow the scalability.

Experiments show that our approach has a good response time and the size of the created index is improved because of the storing of necessary information and the use of MapReduce.

Our approach is only a first phase in a global work of a research about new techniques for querying web data sources.

To a continuation of our work, we will integrate our approach in a global system to exploit our indexes and test relevance of responses that will be returned to users queries.

References

[1] http://searchcloudcomputing.techtarget.com/definition/Hadoop.

[2] J. Dean and S. Ghemawat. Mapreduce: Simplified data Processing on large clusters. OSDI '04, pages 137–150, 2008.

[3] M.Isard, M.Budiu, Y.Yu, A.Birrell, and D.Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pages 59–72. ACM2007.

[4] S. Agrawal et al. Database Tuning Advisor for Microsoft SQLServer2005. VLDB2004. Pages 1110–1121. 2004.

[5] N. Bruno and S. Chaudhuri. Physical Design Refinement: The Merge-Reduce Approach.ACM TODS, 32(4), 2007.

[6] S.Chaudhuri and V. R. Narasayya. Self-Tuning Database Systems: A Decade of Progress. In VLDB, pages 3–14, 2007.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach,M.Burrows,T.Chandra A.Fikes, and R.E. Gruber, "Bigtable: A distributed storage system for structured data" ACM Trans. Comput. Syst. vol. 26, no. 2, pp. 1–26. 2008.

[8] J. Lin, D. Ryaboy, and K. Weil.Full-text indexing for optimizing selection operations in large-scale data analytics. In MapReduce, pages 59–66, 2011.

[9] Mustapha Baziz, Mohand Boughanem, Salam Traboulsi: A concept-based approach for indexing documents in IR. INFORSID 2005: 489-504.

[10] Mustapha Baziz, Mohand Boughanem, and Nathalie Aussenac-Gilles: Conceptual Indexing Based on Document Content Representation. CoLIS 2005: 171 - 186.

[11] Mustapha Baziz, Mohand Boughanem, Gabriella Pasi, Henri Prade: An Information Retrieval Driven by Ontology: from Query to Document Expansion. RIAO2007.

[12] Gerard Salton: Syntactic Approaches to Automatic Book Indexing. ACL 1988: 204-210.

[13] Christina Lioma, Iadh Ounis: Light Syntactically-Based Index Pruning for Information Retrieval. ECIR2007:88-100.

[14] Lin, J. et C. Dyer (2010).Data-Intensive Text Processing with MapReduce.Morgan & Claypool Publishers.

[15] C., Schenk, S., Scherp, A.: Kat: the k-space annotation tool. In: Poster Session, Int. Conf. on Semantic and Digital Media Technologies (SAMT). Germany. (2008).

[16] Russell, B., Torralba, A., Murphy, K., Freeman, W.: Labelme: A database and web-based tool for image annotation. International Journal of Computer Vision 77 (2008) 157-173.

[17] Kipp, M.: Anvil - a generic annotation tool for multimodal dialogue. In: in Proc. 7th European Conf. on Speech Communication and Technology (Eurospeech), Aalborg, Denmark. (2001).

[18] Schallauer, P., Ober, S., Neuschmied, H.: Efficient semantic video annotation by object and shot re detection. In: Posters and Demos Session, 2nd International Conference on Semantic and Digital Media Technologies (SAMT), Koblenz, Germany. (2008).

[19] Kang, B.-Y., & Lee S.-J. "Document indexing: a Concept-based approach to term weight estimation." Information Processing and Management: an International Journal 41(5): 1065 – 1080. (2005).

[20] Nick Craswell, Stephen E. Robertson, Hugo Zaragoza, Michael J. Taylor: Relevance weighting for query independent evidence. SIGIR 2005: 416-423 (2005).

[21] Miller F., WordNet: A lexical database. Communication of the ACM, 38(11): 39-41, (1995).

[22] G. Wang, Evaluating Mapreduce system performance: A Simulation approach. Ph.D. Thesis. Virginia Polytechnic Institute and State University (2012)

Authors' Profiles

Saidi Imene: PhD candidate in computer science, at Litio Laboratory, University of Oran Es Senia, Algeria. She received 5-year engineering degree in computer science in 2010 and a Master Degree in 2011. Her research interests are data mining, information retrieval and web technology.

Nait Bahloul Safia: Associate professor in department of computer science, University of Oran Es Senia, Algeria. Since 2011, she has been leading a team on the topic of data engineering and Web Technology. Her research covers advanced aspects of databases, Web technology and unsupervised classification.

How to cite this paper: Saidi Imene, Nait Bahloul Safia,"An Approach for Indexing Web Data Sources", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.9, pp.52-58, 2014. DOI: 10.5815/ijitcs.2014.09.07

