Ontology-based approach for unsupervised and adaptive focused crawling

Thomas Hassan*
thomas.hassan@u-bourgogne.fr
Le2i FRE2005, CNRS, Arts et Métiers,
Univ. Bourgogne Franche-Comté
Dijon, France

Christophe Cruz
christophe.cruz@u-bourgogne.fr
Le2i FRE2005, CNRS, Arts et Métiers,
Univ. Bourgogne Franche-Comté
Dijon, France

Aurélie Bertaux
aurelie.bertaux@u-bourgogne.fr
Le2i FRE2005, CNRS, Arts et Métiers,
Univ. Bourgogne Franche-Comté
Dijon, France

ABSTRACT

Information from the web is a key resource exploited in the domain of competitive intelligence. These sources represent important volumes of information to process everyday. As the amount of information available grows rapidly, this process becomes overwhelming for experts. To leverage this challenge, this paper presents a novel approach to process such sources and extract only the most valuable pieces of information. The approach is based on an unsupervised and adaptive ontology-learning process. The resulting ontology is used to enhance the performance of a focused crawler. The combination of Big Data and Semantic Web technologies allows to classify information precisely according to domain knowledge, while maintaining optimal performances. The approach and its implementation are described, and an presents the feasibility and performance of the approach.

CCS CONCEPTS

•Information systems → Web searching and information discovery; Semantic web description languages; *Similarity measures; Relevance assessment;* **•Computing methodologies → Parallel computing methodologies; Machine learning; •Computer systems organization → Parallel architectures;**

KEYWORDS

Focused Crawler ; Ontology ; Classification ; Reasonning ; Adaptive systems ; Cross-Referencing

ACM Reference format:
Thomas Hassan, Christophe Cruz, and Aurélie Bertaux. 2017. Ontology-based approach for unsupervised and adaptive focused crawling. In *Proceedings of SBD'17, Chicago, IL, USA, May 19, 2017,* 6 pages.
DOI: http://dx.doi.org/10.1145/3066911.3066912

1 INTRODUCTION

Everyday in the context of competitive intelligence, experts look for information in a high number of information sources to redirect important information to their clients. An important part of their work consists in gathering, analyzing, and summarizing information. This process is done manually by experts on a daily basis. As the number of sources evolves rapidly, this process becomes overwhelming. In particular, Cross-Referencing[1] is an important aspect of this process, required to synthetize information, as well as verify its Veracity. This task is even more challenging in an uncontrolled web environment where content is dynamic, and constantly evolving: the data is nonstationary (i.e. evolving/drifting), where the probabilistic properties of the data change over time.

Considering these challenges, [1](to-be published) presents a novel architecture called SemXDM to perform cross-referencing of web items without overloading the experts. Unlike previous approaches, the system is adaptive and scalable: a scalable ontology-based classification method based on web-reasoning is used, while ontology evolution is managed at crawling time. The system is designed as a set of modules, which revolve around an ontology-described knowledge base. Five distinct modules compose the process, i.e. Recommender, Crawling, Classification, Maintenance, and Priority modules.

To the extent of our knowledge, a scalable and adaptive focused crawling process based on unsupervised ontology-learning and web-reasoning is novel. This paper extends the work of [1] with additional information regarding the Priority module, and the architecture's implementation. An evaluation of the architecture is also presented, evaluating the feasibility and performance of the approach. Next section discusses related work in web crawling, focusing on the use of ontologies in this context. The third section describes the system architecture and each of the modules. The fourth section presents the system implementation and a set of evaluations to validate the approach. The last section concludes and draws lines for current and future work.

2 RELATED WORK

The objective of focused crawlers, or topic-oriented crawlers, is to limit the crawling process only to pages relevant to the topic. To optimize chances to find relevant pages, several sub-types of focused crawlers have emerged.

Learning crawlers can learn on the topic from a set of example pages (training set). Training may also involve learning the path leading to relevant pages. Early approaches in designing learning crawlers use classifiers such as Nave Bayesian classifier to distinguish relevant pages [2]; others suggest using decision trees [5], First Order Logic [14], Neural Networks and Support Vector Machines [7].

These models are based on lexical term matching, thus they do not take into account the semantics associated to the text. Semantic

*The corresponding author

SBD'17, Chicago, IL, USA

DOI: http://dx.doi.org/10.1145/3066911.3066912

[1] Cross-referencing consists in finding multiple sources for a same piece of information.

crawlers resolve this issue using term taxonomies or ontologies to describe documents. In these approaches, ontologies and/or taxonomies are used to enhance the description of documents. Document similarity is then computed by VSM or by specialized models such as the Semantic Similarity Retrieval Model (SSRM) [12].

The main limitations of ontology-based (semantic) crawlers is that the crawler's performance is highly dependant on the ontology used. Two notable issues emerge [4]: as ontologies are designed by domain experts, their understanding of the domain knowledge and the domain knowledge that exists in the real world can be different. Secondly, knowledge described in the data is dynamic and is constantly evolving. These changes in the data distribution can induce more or less radical changes in the target concept, which is generally known in literature as concept drift [8]. In nonstationary data distribution, the performance of a non-adaptive classification model trained under the false stationary assumption may degrade the classification performance over time.

Because of these issues several approaches are interested in integrating semantic-based crawling techniques with ontology learning techniques. [16] proposes a supervised ontology-learning-based focused crawler that aims to maintain the harvest rate of the crawler in the crawling process. The main idea of this crawler is to construct an artificial neural network (ANN) model to determine the relatedness between a Web document and an ontology.

[10] describes an unsupervised ontology-learning-based focused crawler in order to compute the relevance scores between topics and Web documents. The relevance score between a Web document and the topic is the weighted sum of the occurrence frequencies of all the concepts of the ontology in the Web document.

[4] proposes a self-adaptive semantic focused (SASF) crawler, by combining the technologies of semantic focused crawling and ontology learning. Ontology learning technology is used to maintain the high performance of the crawler in the uncontrolled Web environment.

Based on the litterature review, two limitations of focused crawlers have been identified. First, few approaches combine web content mining and web graph mining to harness the web efficiently. Secondly, ontology-based approaches have advantages but often fail to adapt to non-stationary data such as the web, where content is dynamic and constantly evolving.

To tackle these challenges, a novel web mining system called SemXDM (Semantic Cross-Referencing Data Mining) is proposed, extending the approaches of [9][8]. This process uses a focused crawler and a semantic-based classifier to cross-reference data items without expert intervention. Next section describes the SemXDM system and details each of its components. More details on the architecture can be found in to-be published approach described in [1].

3 SEMXDM: SYSTEM DESCRIPTION

SemXDM performs cross-referencing of data items at web scale, using Semantic Web and Big Data technologies. An item is any type of web document which contains text, including (but not restricted to) web pages. As in [9], an ontology-described knowledge base is used to describe, and classify the items. This ontology has two purposes: first, it serves as a classification model, composed of a label hierarchy taxonomy, along with classification rules, as exposed in the approach of [9]. Secondly, data items (*Item* class) are integrated in the ontology at the assertion level, thus the ontology is a repository from which data items can be retrieved.

Figure 1 describes the system and the role of each module. The next subsections describe each of the system's modules individually.

3.1 Recommender Module

The recommender module interacts with the user and respond to its queries. The user can issue two types of queries : Item and Cross-reference queries.

Item Queries searches for items according to an input query in the form of a set of Terms. The search for relevant items is performed offline, ie. all returned items are integrated in the ontology-described knowledge base. The set of Terms $\omega_{term_i} = (term_1, term_2, ..., term_n)$ as is used as input. The output consists of the set of items $\omega_{item_j} = (item_1, item_2, ..., item_n)$ where $|\{\forall j, \exists item : hasTerm(item_j, term_i) \wedge term_i \in \omega_{term_i}\}|$. All items in the set ω_{item_j} can then be ranked using Vector Space Model and similarity measures such as cosine similarity[13].

Cross-reference queries take a classified document in the ontology described knowledge base as input. The query is sent to the crawling module, which searches for similar items on the web: relevance of discovered items by the crawling module is computed as the similarity between the initial item (query) and newly discovered items, as described in section 3.4.

3.2 Crawling Module

The crawling module uses one or multiple web crawlers to mine the web, searching for relevant items. It is triggered when a cross-referencing query is received from the recommender module. Web crawling is an iterative process: at each step, a fixed set of unvisited

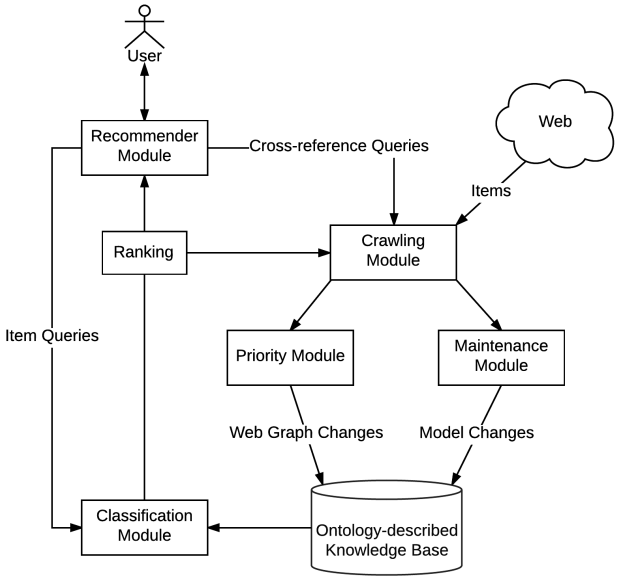


Figure 1: SemXDM components

URLs, called the frontier, is considered for future fetching[2]. The most promising URLs are fetched, which originates a new frontier to be explored in the next step.

The crawling process is parrallelized, as exploring huge portions of the web is a time-consuming task for one machine.

An item is classified based on relevant terms in its content. A Term-based inverted index is used to store new data items and extract relevant terms for classification. After each crawling step, the Term-based inverted index is updated with all newly acquired items from the web. The index allows efficient retrieval of items by Term. The inverted index is composed by a set of item vectors, one vector for each term ∈ Term, in the form: $v_{term} < item_1, item_2, ..., item_n >$. Terms are extracted from each item in the form $v_{item_i} < term_1, term_2, ..., term_n >$ using a term extraction approach proposed in [9], which includes spelling correction, stop-word and synonym detection. Labels corresponds to the most relevant terms, as described in [9], with $Label \sqsubseteq Term$.

3.3 Classification Module

This module performs the classifiction of new items at crawling time, using a Hierarchical Multi-Label Classification method described in [9]. In this approach, relevant terms are first extracted from items as Labels (*Label* class in the ontology). Then, a label hierarchy and classification rules are constructed from the whole item set. Finally, a web-reasoner is used to classify new items according to the label hierarchy and classification rules.

Each item is described with a set of relevant terms. Term relevance is determined based on the indiviual tf-idf value of each term in the set of crawled items. To infer the most specific labels, the rule generation process described in [9] is used : based on the extracted set of terms, rule-based reasoning applies exhaustively a set of rules to a set of triples (i.e. the extracted terms) to infer conclusions [11], i.e. the item's classifications or labels. The rules are translated into rules in the Semantic Web Rule Language (SWRL) and integrated in the ontology to perform the classification.

Classification at query-time is more suited in the context of this work, because items must be re-classified as the ontology-described knowledge base evolves (see next section). Thus, a backward-chaining inference engine is used to determine the classifications of each item at query-time.

After an item is successfully classified with terms and labels, it is possible to determine its relevance compared to the initial query (i.e. item or cross-reference query).

The result is a label vector $v^{label}_{item_i}$, where each dimension is a *Label* in the ontology. The label vector is used in the priority assignement method (next section).

As described in previous section, focused crawlers lack ways to adapt to the web environment where data is non-stationary. The classifier used in a focused crawler must adapt to a constantly changing environment, where features are evolving[4]. Secondly, focused crawlers should learn from experience how to find relevant information, using the graph of the web (hyperlinks) [3]. To leverage theses issues, a Maintenance module and a Priority module are described in the next sections. The objective of the Maintenance module is to adapt the ontology-described knowledge base over time. The Priority module learns ways to find relevant information, by enhancing the priority assignment of items.

[2]fetching is the action of downloading a web page to extract its content

3.4 Maintenance Module

[8] proposes a scalable Adaptive Learning process to consistently adapt an Ontology-described classification model used for hierarchical multi-label classification regarding a non-stationary stream of unstructured text data in Big Data context. This approach shows enhancement of the classifier's performance when the classification model is updated with a stream of data.

The crawling module outputs new data items, which are used as the input of the maintenance process. The Maintenance Module integration is depicted in figure 2.

The adaptive learning process focuses on adapting the ontology-described classification model regarding a stream of unstructured text documents in Big Data context. Three characteristics of data stream processing are addressed in the adaptive learning process: feature-evolution, concept-evolution, and concept-drift [6]. Using the approach described in [8], the adaptive learning process manages the impact of changes in the classification model according to the specific semantic of the classification process. The approach is based on a cyclic maintenance scheme [15] to constantly update a cooccurrence matrix of the features (terms) retrieved from the data stream. Changes in the conditional proportion of terms are detected, and their impact on the classification model are propagated.

As the process uses classification at query-time to classify items, the evolution of the ontology-described classification model impacts automatically the classifications of new items.

3.5 Priority Module

The objective of the Priority Module is to combine web graph information and content information retrieved on the web to enhance and guide the search of the Crawling Module, by defining the priority of the link frontier. The priority module computes expected

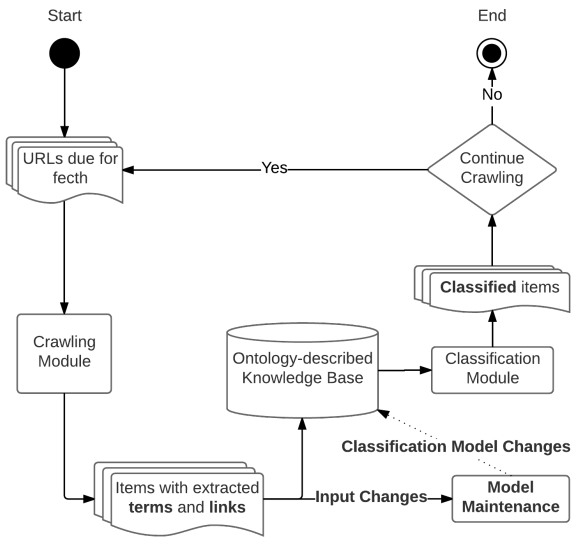


Figure 2: Maintenance Module

scores of items in the frontier, i.e. items that have yet to be downloaded. The Priority Module update web graph information after each mining operation (crawl) in the ontology, as described in figure 3. The webgraph is composed of the complete set of discovered items, along with web graph information, i.e. outlinks and inlinks between items.

For each data item $item_i$, the Priority Module populates the ontology-described knowledge base with new inlinks and outlinks at the assertion level. The *hasInlink* and *hasOutlink* properties define web graph information for each item in the ontology-described knowledge base.

To identify potential paths leading to relevant items, the context graph approach described in [3] is used. After each crawl, the different layers of the context graph are recomputed as described below.

A weighted vector v_i^{term} for each layer L_i, where each dimension of the vector is a *Term*. Each vector $v_{item_i}^{term}$ is recomputed after each crawl, as new relevant items and links are discovered and added to the web graph. A second vector $v_{item_i}^{label}$ is generated, such as each dimension is a *Label* in the ontology.

The degree of belonging of an item in layer L_i is then approximated by calculating the distance between the term and label vectors, respectively with the term vector and label vector extracted from new items, as defined in section 3.3. The cosine similarity between each pair of vectors is used for that purpose, such as:

$$cos(\boldsymbol{v}_{L_i}^{term}, \boldsymbol{v}_{item_j}^{term}) = \frac{\boldsymbol{v}_{L_i}^{term} \cdot \boldsymbol{v}_{item_j}^{term}}{||\boldsymbol{v}_{L_i}^{term}|| \cdot ||\boldsymbol{v}_{item_j}^{term}||} \quad (1)$$

The similarity calculation of the labels vectors is identical. Both vectors are used to determine the similarity of items with each layer. We define the degree of belonging d_i for each layer of the graph as the arithmetic mean of label and term similarity as follows:

$$d_i = \frac{cos(\boldsymbol{v}_{L_i}^{term} \boldsymbol{v}_{item_j}^{term}) + cos(\boldsymbol{v}_{L_i}^{label} \boldsymbol{v}_{item_j}^{label})}{2}) \quad (2)$$

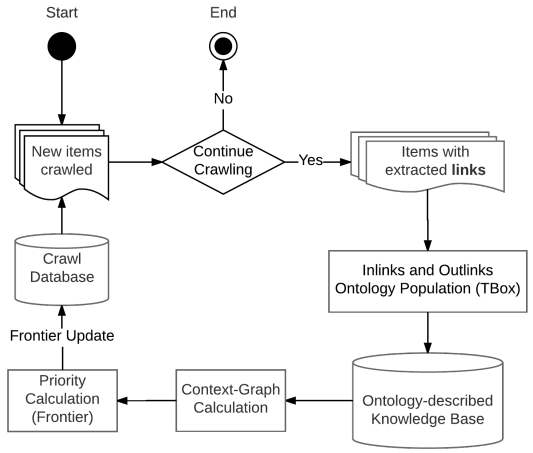


Figure 3: Priority Module

As some items might not belong to label classes from the predictive model, we assume that term similarity is a more reliable value, thus we define the final **relevance score** as follows:

$$r_{item_j} = max(cos(\boldsymbol{v}_{L_i}^{term} \boldsymbol{v}_{item_j}^{term}), d_i) \quad (3)$$

where $r_{item_j} \in [0, 1]$. This ensures that items with few labels but high term similarity are still ranked highly. Another solution would be to weight labels and terms depending on the use case.

The resulting score is used to approximate the distance from a page to a relevant page, and re-rank the frontier according to the new web graph after each crawl. Data items that are potentially closer are have their priority in the frontier queue increased based on their belonging in the context-graph : the outlinks extracted from the most similar items are prioritized in the frontier queue. Each new item is assigned to the most appropriate layer, using the **relevance score** as scoring function. After each crawl cycle, the context graph is updated. A relevance threshold θ is defined by the user. The set of highly relevant items $\omega_{L_0}^{rel}$ is determined, such as all items in the set have a relevant score for the first layer of the graph L_0, i.e. $r_{item_i} > \theta$. In the first crawling step, this set corresponds to user-provided items, ie. seed items.

4 IMPLEMENTATION, EVALUATION

This section presents the system's implementation and two evaluations of the approach. The first is a proof-of-concept, which aims to validate the approach's feasability. The second is a quality evaluation that monitors the performance of the approach for information retrieval(cross-referencing).

4.1 Implementation

The system is implemented as a Java application and deployed on a Hadoop Cluster[3], which integrates the different modules with several scalable tools from the Semantic Web and Big Data domain. All nodes of the cluster are equipped with Intel Core I5 Quad Core processors and 8GB of RAM, except for the master node, equipped with 16GB. The Stardog triple-store is deployed on a dedicated server with 8GB of RAM. Apache Nutch[4] is used a scalable and distributed crawler. The maintenance module is based on Apache Storm and Apache Kafka[5] as described in [8]. Derived classes from the Nutch API allows to integrate the other modules. Apache Solr[6] is used to build the inverted index of crawled items. Finally, Stardog[7] is used as a scalable triplestore to integrate the ontology-described knowledge base.

4.2 Proof of concept

4.2.1 Setup. This evaluation consists in monotoring each module's behaviour during a crawling process following a cross-reference query. An economic article is used as the cross-reference query. A training dataset composed of 45k economic-related news is used to train the classification model prior to crawling. Also, a list of 1200 economic-related URL seeds is used as the starting point of the

[3] http://hadoop.apache.org/
[4] http://nutch.apache.org/
[5] http://storm.apache.org/, https://kafka.apache.org/
[6] http://lucene.apache.org/solr/
[7] http://stardog.com/

crawler. The priority module consists of a 3-layer graph (layers 0 to 2). Layer 0 is initially composed of one item to be cross-referenced, while layers 1 and 2 are empty. The relevance threshold θ for the first layer is set to 0.5, i.e. items for which the relevance score with the initial item is greater than 0.5 will be added to the layer 0 of the graph. This value is intentionnally restrictive to limit the size of the context-graph and restrict the search to highly similar items.

4.2.2 Results. The data obtained from the crawl is divided by crawl iteration, or "round" : each round, the top 2000 ranked items in the frontier queue are fetched. The total number of fetched items is approximately 15000 over 9 rounds as some links naturally fail to be fetched. Figure 4 describes the number of labels inferred (i.e. classifications) at each step of the crawl.

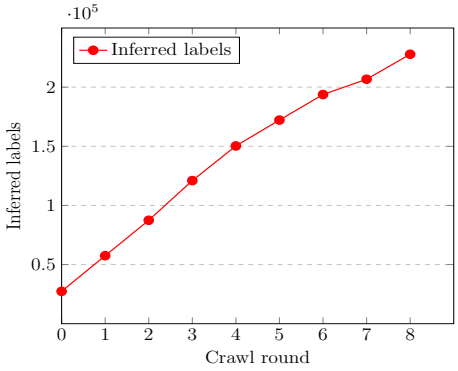


Figure 4: Inferred labels (classification module)

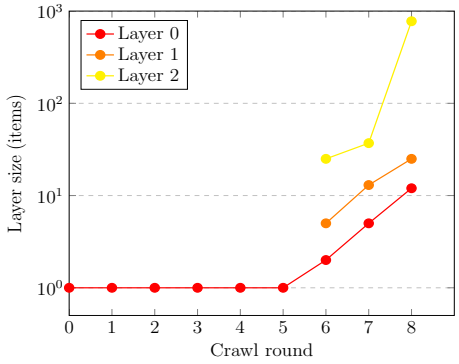
The number of classifications increases accordlingly to the number of items classified and to the increase of the number of rules (figure 5.c).

Figure 5.a shows the increase in the number of items per layer at each step. We can observe that no relevant item is added to the layer 0 before step 5, hence all layers only increase from step 6. A more lenient (lower) value of the θ threashold would make the context-graph grow faster, while decreasing its quality, i.e. its similarity with the initial item.

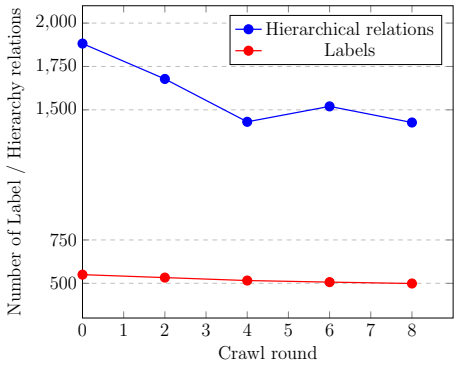
Finally, figure 5.b and 5.c shows the evolution of the classification model according to the detected input changes. Values for the round 0 are the initial properties of the classification model, generated from the initial dataset (45k items). Similarly to the observations in [8], most changes in the classification model are additions of new classifications rules, although as the item set is more diverse (items are crawled from very diverse web sources), the ratio of learned terms and rules is higher. A limitation of the process is the high computation cost of the mainetnance process. The total number of term coocurrences in one item can be very high depending on its dimension, thus the propagation of these input changes exceeds the average time of all the other modules. The technical limitations of the experiment environment makes it hardly practicable in real time.

4.3 Quality evaluation

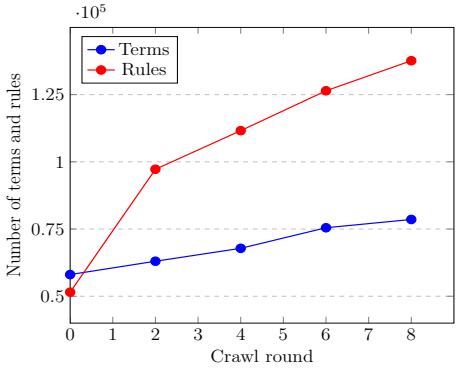
4.3.1 Setup. This evaluation consists in evaluating the performance of the approach using standard crawling metrics, i.e. average



(a) Context-graph size (priority module)



(b) Label hierarchy evolution



(c) Terms and rules evolution

Figure 5: Context-graph and classification model evolution

similarity and harvest rate. Parameter values and initial setup remain unchanged. The term-based cosine similarity of the retrieved items against the first layer of the context graph (L_0) is used as the similarity score. The harvest rate is a user-defined metric which represents the proportion of relevant items over the total number of crawled items. In our study we define the harvest rate as the set of items for which the term-based similarity is greater than 0.25, i.e. $cos(\boldsymbol{v}_{L_0}^{term}, \boldsymbol{v}_{item_j}^{term}) > 0.25$.

We compare the performance of the proposed approach with an implementation of a best-N-first crawler, against the same cross-reference query. The term-based cosine similarity and harvest rate are used to compare both approaches : the objective of each algorithm is thus to maximize both metrics.

4.3.2 Results. Figure 6 presents the comparison between the SEMXDM approach and a Best-N-First crawler for a cross-reference query (an input economic article).

The results show that for a sample cross-reference query, the SEMXDM approach performs better than a standard Best-N-First approach using standard metrics. In particular, the harvest rate data shows that the proportion of highly relevant items is significantly higher in SEMXDM after 5 crawl rounds, which is an important factor for the cross-referencing task. This indicates that using the **relevance score** as a priority function using both term and label similarity instead of standard cosine similarity improves the performance of the crawler. These are still early results, as the evaluation of the architecture represents ongoing work. It is required to perform a larger scale cross-reference evaluation, averaging the performance of the approach over a high number of cross-reference queries. Secondly, we consider comparing the approach with a wider range of focused crawling approaches, and analyzing the individual impact of each module on the performance of the approach to refine the results and optimize each module.

5 CONCLUSION

In this paper we present a novel approach for unsupervised focused crawling, and a preliminary evaluation to validate its implementation. This approach, called 'SemXDM', tackles several issues in focused crawling and performs cross-referencing of web documents using Semantic Web and Big Data techonologies. Classification of web items at crawling time is achieved using a scalable ontology-based approach. Web graph mining is used in conjunction with an adaptive maintenance process to increase the efficiency of the cross-referencing process.

The system has been implemented with success and deployed on a Big Data platform. A preliminary evaluation shows that the ontology-described classification model performs classification of web sources at crawling time. Although the adaptive maintenance process allows tho adapt the classification model to new data in real time, technical limitations of the approach underline the high computation cost of the maintenance process. The quality evaluation shows promising performances for a sample cross-reference task, but larger scale tests are required to assess the performance of the approach.

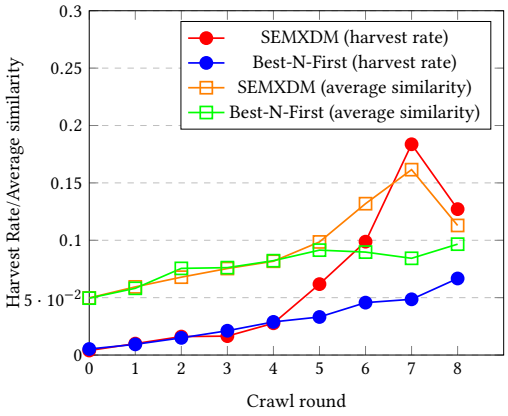


Figure 6: Average term-based cosine similarity and harvest rate per round(cross-reference query)

ACKNOWLEDGMENTS

This project is funded by the company Actualis SARL, the French agency ANRT, the Bourgogne Franche Comté region and the European Regional Development Fund (ERDF).

REFERENCES

[1] Anon. Forthcoming 2017. Predictive and Evolutive Cross-Referencing for Web Textual Sources. In *SAI Computing Conference (SAI), 2017*. ACM.
[2] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. 1999. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks* 31, 11 (1999), 1623–1640.
[3] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C Lee Giles, Marco Gori, and others. 2000. Focused Crawling Using Context Graphs.. In *VLDB*. 527–534.
[4] Hai Dong and Farookh Khadeer Hussain. 2014. Self-adaptive semantic focused crawler for mining services information discovery. *IEEE Transactions On Industrial Informatics* 10, 2 (2014), 1616–1626.
[5] Jun Li, Kazutaka Furuse, and Kazunori Yamaguchi. 2005. Focused crawling by exploiting anchor text using decision tree. In *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM, 1190–1191.
[6] Mohammad M Masud, Qing Chen, Latifur Khan, Charu Aggarwal, Jing Gao, Jiawei Han, and Bhavani Thuraisingham. 2010. Addressing concept-evolution in concept-drifting data streams. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 929–934.
[7] Gautam Pant and Padmini Srinivasan. 2005. Learning to crawl: Comparing classification schemes. *ACM Transactions on Information Systems (TOIS)* 23, 4 (2005), 430–462.
[8] Rafael Peixoto, Christophe Cruz, and Nuno Silva. 2016. Adaptive learning process for the evolution of ontology-described classification model in big data context. In *SAI Computing Conference (SAI), 2016*. IEEE, 532–540.
[9] Rafael Peixoto, Thomas Hassan, Christophe Cruz, Aurélie Bertaux, and Nuno Silva. 2015. Semantic HMC: a predictive model using multi-label classification for big data. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, Vol. 2. IEEE, 173–179.
[10] Chang Su, Yang Gao, Jianmei Yang, and Bin Luo. 2005. An efficient adaptive focused crawler based on ontology learning. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*. IEEE, 6–pp.
[11] Jacopo Urbani, Frank Van Harmelen, Stefan Schlobach, and Henri Bal. 2011. QueryPIE: Backward reasoning for OWL horst over very large knowledge bases. In *Lecture Notes in Computer Science (ISWC'11)*, Vol. 7031 LNCS. Springer-Verlag, Berlin, Heidelberg, 730–745. DOI: http://dx.doi.org/10.1007/978-3-642-25073-6{_}46
[12] Giannis Varelas, Epimenidis Voutsakis, Paraskevi Raftopoulou, Euripides GM Petrakis, and Evangelos E Milios. 2005. Semantic similarity methods in wordNet and their application to information retrieval on the web. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*. ACM, 10–16.
[13] David Werner and Christophe Cruz. 2013. A method to manage the precision difference between items and profiles: in a context of content-based recommender system and vector space model. In *Signal-Image Technology & Internet-Based Systems (SITIS), 2013 International Conference on*. IEEE, 337–344.
[14] Qingyang Xu and Wanli Zuo. 2007. First-order focused crawling. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 1159–1160.
[15] Fouad Zablith, Grigoris Antoniou, Mathieu d'Aquin, Giorgos Flouris, Haridimos Kondylakis, Enrico Motta, Dimitris Plexousakis, and Marta Sabou. 2015. Ontology evolution: a process-centric survey. *The knowledge engineering review* 30, 01 (2015), 45–75.
[16] Hai-Tao Zheng, Bo-Yeong Kang, and Hong-Gee Kim. 2008. An ontology-based approach to learnable focused crawling. *Information Sciences* 178, 23 (2008), 4512–4522.