S. Albitar, B. Espinasse, S. Fournier (2010), « Combining Agents and Wrapper Induction for Information Gathering on Restricted Web Domains », IEEE-RCIS 2010, Fifth International Conference on Internet and Web Applications and Services, Nice, France, May 19-21, 2010.

Combining Agents and Wrapper Induction for Information Gathering on Restricted Web Domains

Shereen Albitar
LSIS UMR CNRS 6168
Université d'Aix-Marseille,
Domaine Universitaire de St Jerôme,
F-13997, Marseille Cedex 20, France
shereen.albitar@lsis.org

Bernard Espinasse
LSIS UMR CNRS 6168
Université d'Aix-Marseille,
Domaine Universitaire de St Jerôme,
F-13997, Marseille Cedex 20, France
bernard.espinasse@lsis.org

Sébastien Fournier
LSIS UMR CNRS 6168
Université d'Aix-Marseille,
Domaine Universitaire de St Jerôme,
F-13997, Marseille Cedex 20, France
sebastien.fournier@lsis.org

Abstract — **Web is growing constantly and exponentially every day. Thus, gathering relevant information becomes unfeasible. Existent indexing-based search engines ignore information context, which is essential to deciding on its relevance. Restraining to a single web domain, domain ontology can be used to take into consideration the related context, the fact that might enable treating web pages that belong to the considered domain more intelligently. Nevertheless, symbolic rules that exploit domain's ontology to realize this treatment are delicate and fastidious to develop, especially for information extraction task. This paper presents Boosted Wrapper Induction (BWI), a machine learning method for adaptive information extraction, and its exploitation as a replacement of the symbolic approach for information extraction task in AGATHE, a generic multi-agent architecture for information gathering on restrained web domains.**

Keywords-component; Information extraction; machine learning; multi-agent systems; boosted wrapper induction; cooperative information gathering.

I. INTRODUCTION

The growing size of the web and the heterogeneity of accessible pages made information gathering more and more complex. In order to retrieve relevant information, gathering must be restrained to specific web domains, in other words, the context in which information is gathered must be taken into consideration.

Considering context permits a better treatment of information contained in collected web pages. For instance, first of all these web pages can be classified according to the different classes specific to the concerned web domain, then, relevant information can be extracted from pages belonging to the same class more accurately. Most information extraction from web pages belonging to the same class is realized in this way (for example, researcher's interesting subjects, etc.).

This work has been realized within the framework of the project AGATHE [1] that proposes a generic multi-agent architecture for contextual information gathering on restricted web domains. In AGATHE, software agents exploit ontologies in order to realize web page classification and information extraction tasks. This paper is focused particularly on the information extraction (IE) task.

IE consists in extracting automatically relevant information from text. Many software systems were developed to accomplish IE tasks. Earlier, these systems were mainly based on symbolic rules handcrafted by domain experts [2, 3]. Considering the dynamism and the heterogeneity of the web, they should be constantly modified and maintained. This constant maintenance is time consuming and requires expertise in the application domain.

Recently, progress achieved in the domain of supervised and non-supervised *Machine Learning* (ML) algorithms has simplified the development of IE programs; these algorithms have been utilized to automate rule production. This evolution was the origin of *Adaptive Information Extraction* (AIE). In recent times, many AIE systems had been developed following three-step procedure: (1) Recognizing semantically relevant information in the text (2) Extracting this information (3) Stocking it in an organized structure or in a database for future analysis [2, 4].

Currently in AGATHE, first web pages are retrieved from the web. Then, some agents use domain ontology to classify them semantically. Finally, other agents, depending on the same ontology, extract relevant information from these pages according to their classes. Based on symbolic rules (production rules) handcrafted in Jess platform, the implementation of these tasks is painful especially for the IE task.

The goal of this work was firstly to carry out a study of different IE techniques using ML algorithms (AIE), secondly, to replace, in AGATHE, information extraction symbolic rules by AIE techniques encapsulated in software agents, leaving the semantic classification task intact; it is still implemented in symbolic rules deploying domain's ontology. Coupling both symbolic and machine learning techniques in AGATHE is the major contribution of this work leading to improve its IG capabilities.

In the second section, we study the advantages of using ML techniques in IE (in particular supervised ML) and introduce Adaptive Information Extraction (AIE) with its principal methods. Boosted Wrapper Induction (BWI), the supervised AIE method chosen for this work, is presented in the third section accompanied by a description of the TIES system that was developed in IRST laboratory in Trento implementing this method. Section 4 presents the encapsulation of TIES in a software agent, and then its integration in the new multi- agent architecture of AGATHE. Section 5 presents some implementation details related to the prototype in progress in addition to first results obtained by deploying the new version of AGATHE demonstrating the progress achieved in this work. Finally, we conclude with our research perspectives.

II. MACHINE LEARNING FOR INFORMATION EXTRACTION

Production rule based IE systems used to be developed in an ad hoc manner; they were specific to the application domain for which they had been implemented. In order to easily adapt IE systems to multiple application domains, ML algorithms and statistic methods were deployed [2] leading to Adaptive Information Extraction (AIE).

A. Adaptive Information Extraction (AIE)

Domain specific extraction rules or patterns, instead of being handcrafted for each application domain, can be learnt directly by generic domain-independent IE system using a tagged domain-specific training set [3, 4]. Enriched with natural language processing (NLP) and inductive logic programming (ILP) methods, a spectrum of generic architectures were proposed to realize supervised IE on Web pages [5].

Recently, *self-supervised* IE systems, a new paradigm in IE research domain, has come to life. In general, these systems depend on domain-independent patterns to label their training sets for each application domain [3]. According to the under test performance of self-supervised systems, supervised IE systems are still more attractive. In order to evaluate and compare AIE supervised algorithms, three measures are principally used: *Precision*, *Recall* and *F-Measure* [6].

B. Methods for Adaptive Information Extraction

In the last decade, different classifications for adaptive information extraction methods were proposed [2, 7]. According to the classification proposed in [7], the three distinguished classes of AIE methods are: *Rule Learning based methods, Classification based Methods, and finally, Sequential Labeling based Methods.*

1) Rule Learning based methods

The most widely used among the others, this class includes three different categories of methods:

a) Dictionary based methods

Methods of this type create a dictionary of templates (patterns) using tagged texts and then deploy these templates in extracting relevant information from untagged plain text [7]. In this category, many systems, like AutoSlog [7] and STALKER [8], were developed for IE. Each of them adopts its particular strategy in realizing dictionary learning and IE tasks.

b) Rule based methods

These methods discover the semantic/syntactic characteristics surrounding tagged information in the training set, and then they deploy this knowledge in constructing rules (instead of the pattern dictionary produced by the previous category) for information extraction. In other words they learn the contexts in which relevant information could be probably detected in any text belonging to a specific application domain [7].

Methods that start first by learning special cases and then try to generalize them during training are called bottom-up methods. On the contrary, top-down methods start with a general rule and specialize it according to tagged examples in the training set. We mention particularly in the rule based category the following IE systems: Rapier [9] and LP² [10]. These systems were mostly deployed for extracting information from semi-structured text.

c) Wrapper Induction Methods

This category performs the same steps as the previous one in learning and IE tasks. What makes this category a particular case is that its methods consider HTML tags as well as text in learning extraction wrappers while these tags might cause a great difficulty to other linguistic methods [11]. So additional information provided by HTML tags might help in improving IE results. This is why they were principally used in extracting information from both structured and semi-structured text [5, 7]. In order to cover IE from unstructured text as well, Boosted Wrapper Induction (BWI) algorithm was proposed [11].

2) Classification based Methods

These methods depend on classification algorithms in deciding whether a text partition is relevant to the application domain (is to be extracted) or not [7]. During training phase, a classification-based algorithm is provided with training examples to construct two classification models for both boundaries considering each domain concept. These models are then used to predict and locate relevant information.

Support Vector Machines (SVMs) are largely deployed in developing IE systems in this class, like both ELIE [12] and SIE [13]. Other algorithms, like Maximum Entropy and Voted Perceptron were also the basis of many methods belonging to this class.

3) Sequential Labeling based Methods

In sequential labeling, each token in the document is labeled referring to its characteristics. Deploying interdependencies between different concepts in labeling tokens made sequential labeling a particular case compared to the formerly introduced classes.

Statistical learning algorithms like Markov models and Maximum Entropy models [5], were basically used in developing sequential labeling systems for natural language preprocessing such as Part Of Speech tagging (POS) and also for Information Extraction tasks [2, 7].

III. BOOSTED WRAPPER INDUCTION AND TIES SYSTEM

In our research, we adopted the boosted wrapper induction (BWI) algorithm, a Rule Learning based method belonging to the Wrapper Induction category. The reason for which BWI was chosen for this work is its competence in information extraction from unstructured text in addition to structured and semi-structured text.

We introduce in this section the supervised IE algorithm “BWI” and the generic IE system “TIES” implementing this algorithm.

A. Boosted Wrapper Induction (BWI)

For methods developed on Wrapper Induction paradigm, target documents must have some kind of regularity in their structure, which is not true in most cases.

However, BWI [11, 14] was developed to overcome this limitation enabling IE from unstructured text as well. The main idea was to combine predictions of many simple specific patterns in more general and complex ones, which will have a higher recall rate covering many possible contextual patterns in natural language text. BWI, as in [11], works following this procedure:

```
procedure BWI (example sets S and E)
 {
    F ← AdaBoost(LearnDetector, S)
    A ← AdaBoost(LearnDetector, E)
    H ← field length histogram from S and E
    return wrapper W = <F, A, H>
 }
```

An induced *wrapper* W=<F, A, H> consists of a set of *detectors* that classify the start boundaries of a field (F), another set of *detectors* to classify the end boundaries of the same field (A) in addition to the function H(k) who estimates the prior probability that the specific field has the length k. Each *detector* consists of two *patterns* of tokens surrounding the boundary it classifies.

In order to produce more effective and general patterns, detector learner is wrapped in a *boosting* algorithm (*AdaBoost*) that accomplishes training in a predetermined number of iterations. At the end of every iteration, *AdaBoost* assigns weights to training set examples and confidence values to the induced detectors according to the weights and the number of examples they cover. These weights guide oncoming iterations so uncovered examples are treated more carefully and simple specific detectors are generalized or discarded. This leads to a final group of general detectors with high recall rates and confidence scores.

During IE phase, detectors’ patterns and confidence scores in addition to the field length prediction are combined to classify target field boundaries or in other words to locate and determine tokens of relevant information to extract.

Uses boosting technique to combine many contextual patterns in final effective predictors, BWI is competitive with other IE algorithms, this was the reason for which we’ve chosen this algorithm for this work.

B. The TIES System

TIES, being part of the project IST-Dot.Kom [15] and developed by the IRST of Trento, is a generic modular architecture for supervised IE implementing the formerly detailed algorithm BWI.

Using an annotated training corpus with predefined tags, TIES can automatically generate wrappers (Training phase) which afterwards can be used to extract information from new unannotated text (Extraction phase). Both phases are illustrated in fig. 1.

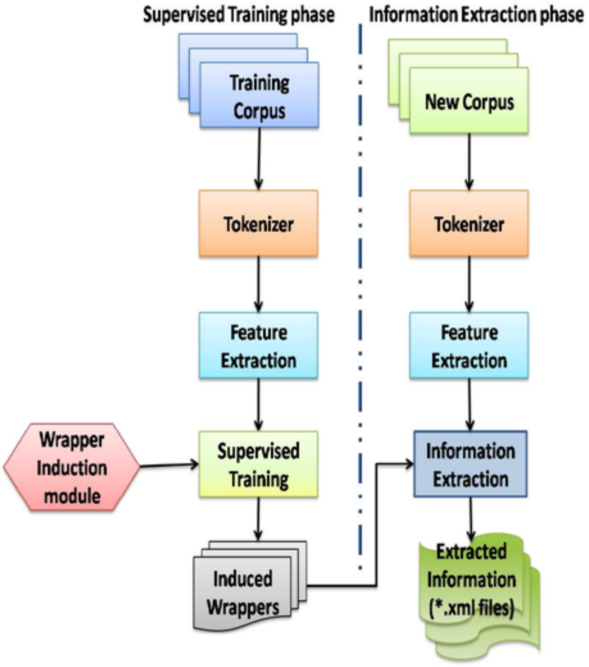

Figure 1. The two phases of TIES’s execution.

1) CFP corpus

The corpus *Call For Papers (CFP)* [16] of the Pascal Challenge was adopted in this work as it had been used earlier as a formal basis to evaluate and compare the performance of different ML algorithms [17]. The majority of the 1100 documents (850 calls for workshops, 250 calls for conferences) constituting the CFP corpus come from the domain of computer science. Others were collected from the biomedicine and linguistics domains.

The 1100 documents were arranged in three different corpuses; the training corpus containing 400 calls for workshops, the test corpus containing 200 calls for workshops and the enrich corpus containing 250 calls for workshops and 250 calls for conferences. A call for workshop might include details regarding conferences as workshops might be related to other conferences.

Corpus’ documents are annotated using eleven tags; eight for workshop concepts and three for conference concepts. Some concepts like dates can be shared between both classes. Conference concepts have lower frequencies in corpus compared to workshop concepts. Table I shows concepts’ frequencies in both training and test corpuses.

TABLE I. CONCEPTS' FREQUENCIES IN CORPUS

Concepts	Frequency			
	Training	*%*	*Test*	*%*
Workshopname	543	11.8	245	10.8
Workshopacronym	566	12.3	243	10.7
Workshophomepage	367	8.0	215	9.5
Workshoplocation	457	10.0	224	9.9
Workshopdate	586	12.8	326	14.3
Workshopsubmissiondate	590	12.9	316	13.9
Workshopnotificationacceptancedate	391	8.5	190	8.4
Workshopcamerareadycopydate	355	7.7	163	7.2
Conferencename	204	4.5	90	4.0
Conferenceacronym	420	9.2	187	8.2
Conferencehomepage	104	2.3	75	3.3
TOTAL	**4583**	**100**	**2274**	**100**

2) *Tokenization step*

In this step, input files (HTML or XML files) are converted into TIES input format (TIESIF); the text is broken down into series of tokens labeled by their identification, type (word, separator or punctuation mark), and position in the original text.

3) *Feature Extraction step*

During this step, a number of Boolean functions are applied to the series of tokens produced through the previous step bringing out attributes to be bounded to each of these tokens. The following is a fragment of the result of tokenization and feature extraction on a document from the CFP corpus. The initial fragment was annotated using the tag (<workshopname>):

```
<token id="27" type="tag"  start="113" len="14"
open_tag="true">workshopname</token>
<token id="28" type="word"  start="127" len="8"
alpha_token="true"
upper_case_token="true">WORKSHOP</token> <token
id="30" type="word"  start="136" len="2"
alpha_token="true"
lower_case_token="true">on</token> <token id="32"
type="word"  start="139" len="10" alpha_token="true"
upper_case_token="true">CONSTRAINT</token> <token
id="34" type="word"  start="150" len="9"
alpha_token="true"
upper_case_token="true">DATABASES</token> <token
id="36" type="word"  start="160" len="2"
alpha_token="true"
lower_case_token="true">in</token> <token id="38"
type="word"  start="163" len="2" alpha_token="true"
upper_case_token="true">AI</token><token id="39"
type="tag"  start="165" len="15"
close_tag="true">/workshopname</token>
```

As the previous example shows, the token *workshopname* is an open tag whereas the token */workshopname* is a closing tag. Also, the tokens *WORKSHOP*, *CONSTRAINT*, *DATABASES*, and *AI* are words in uppercase while *on* and *in* are both in lowercase.

4) *Training step*

The BWI algorithm inducts the wrappers at some point in this step. TIES provide different validation strategies like 4-fold cross validation in order to validate training results; a part of training corpus is reserved to extract information from it using the learnt models to evaluate and validate the results using statistical measures. In fact, this enables evaluating system performance and orients its parameters' calibration.

The next detector is a fore-detector for the beginning of the concept *workshopname*. It indicates that if the three successive tokens pattern (any_token, **Workshop, on**) was matched in text, the beginning of this concept would be detected with a confidence score equal to (**2.209582649580836**).

```
<fore-detector>
<detector>
      <pattern type="prefix" />
      <pattern type="suffix">
        <feature name="any_token" value="true" />
        <feature name="token" value="Workshop" />
        <feature name="token" value="on" />
      </pattern>
      <confidence-
      value>2.209582649580836</confidence-value>
</detector>
</fore-detector>
```

5) *Information Extraction step*

Using the wrappers produced during the training step, target fields can be detected and extracted from new plain text. For each concept, extraction results are organized in a XML file.

A fragment of an output file for the concept *conferenceacronym* is presented below. The element entity contains tokens of the extracted field accompanied by their source and their position in the original text.

```
<entity-list>
<entity name="conferenceacronym"
score="5.125015282673751" src=".\input\CFP\key2\1.5-
train-0-2-3-ESSLLI-LSLCBT_1999.xml" start="89"
end="98" >
       <token start="89" end="7">ESSLLI-</token>
       <token start="96" end="2">99</token>
</entity>
.........
.........
</entity-list>
```

6) *Executing TIES*

Which concepts to learn, which strategy to follow and what value to be assigned to the *lookahead* parameter and other parameters can be passed to TIES using configuration files. This enables the user to control its execution during training phase.

The parameter *lookahead* determines the context that should be taken into consideration throughout training for each detector. In fact, this parameter has a crucial effect on training performance. Assigning the value (3) to the *lookahead*, this means that for every field three tokens before its beginning and three tokens after its beginning should be taken into consideration to learn the fore-detector of its related concept. In other words, the context the algorithm takes into consideration during training phase would cover six tokens.

Fig. 2 illustrates the influence of *lookahead* on the *F-measure* during training. As the value of *lookahead* increases, the value of *F-measure* increases as well for most concepts. Nevertheless, as the value of *lookahead* reaches 5, this increase becomes insignificant and most curves become stable.

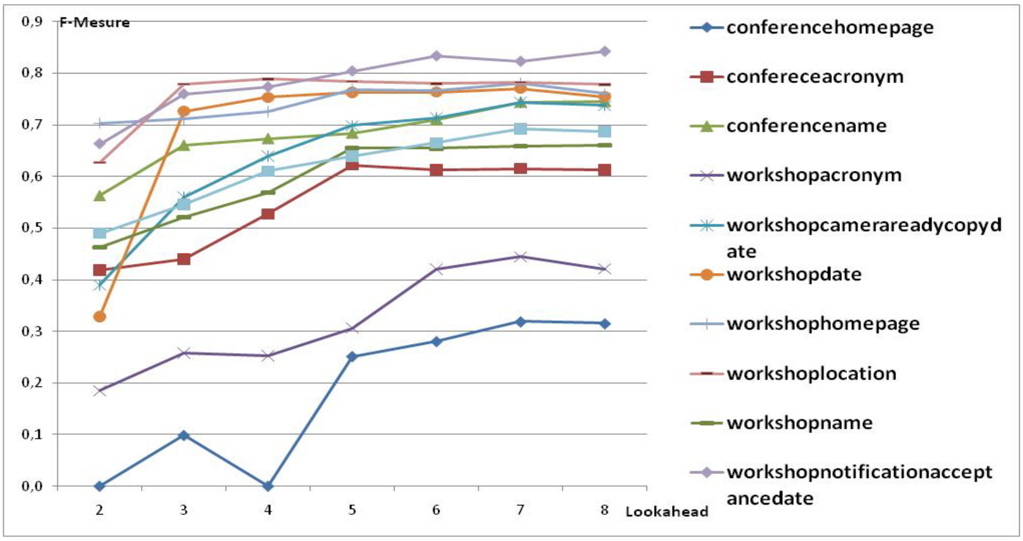

Figure 2. The effect of Lookahead on F-measure

IV. COMBINING AGENTS AND WRAPPER INDUCTION IN AGATHE

In this section, the AGATHE system is briefly presented, and the architecture of its new extraction subsystem, combining agents and wrapper induction techniques, is presented in details.

A. The AGATHE system overview

The AGATHE system is a generic software architecture allowing the development of information gathering systems on the Web, for one or more restricted domains. Being developed between France and Brazil, the system AGATHE [1] implements a cooperative information gathering approach based on software agents that cooperate and exploit ontologies related to restricted web domains.

First version of AGATHE reused the ontology based techniques of classification and extraction associated to these restricted domains of MASTER-Web [18], and deployed them in a complex organization of multiple specialized, effective, and interacting software agents. Moreover AGATHE allows treating several domains of search simultaneously, and deploys mechanisms for inter-domain recommendations.

AGATHE's general architecture [1], illustrated in fig. 3, is particularly composed of three interacting subsystems:

- *The Search Subsystem* is in charge of querying external search engines on the Web (such as Google) in order to obtain Web pages to treat by other subsystems.
- *The Extraction Subsystem* is composed of multiple "extraction clusters" (EC), each of them is specialized in processing web pages of a specific domain (ex. The academic research domain or the tourism domain).
- *The Front Office Subsystem* ensures the storage of the extracted information resulting from web pages treatments in the previous subsystem, and provides a query interface for the users, counting humans and other software agents.

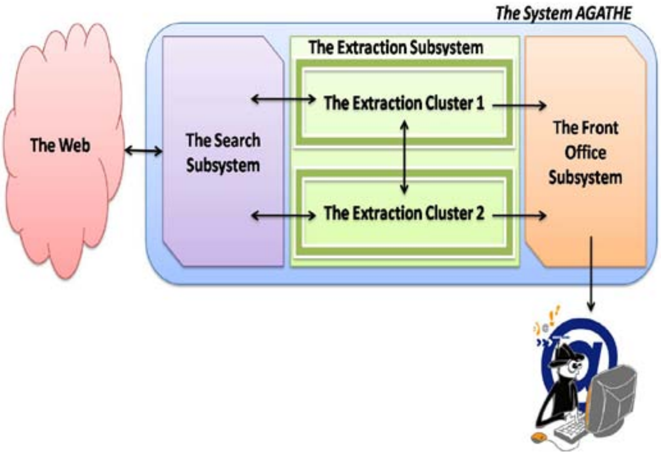

Figure 3. General AGATHE architecture

Software agents using symbolic rules that exploit ontologies realized most tasks in the first AGATHE. Since symbolic rules were domain dependent and arduously written, it seemed judicious to replace a part of them responsible for the information extraction task by TIES, the previously detailed IE system. The aim of this work was to combine symbolic based classification and machine learning information extraction in new extraction subsystem architecture.

Other information gathering systems have also adopted this approach. We mention particularly the system CROSSMARC [19] that was implemented for both e-retail and job offers domains coupling symbolic rules with wrapper induction.

B. AGATHE adaptive extraction subsystem architecture

AGATHE's new extraction subsystem, performing an adaptive information extraction, is composed of extraction clusters. These clusters are composed of software agents performing various tasks, like semantic classification and information extraction, on collected web pages.

The classification task is kept as it was in the first version of AGATHE [1]. Agents that exploit a domain ontology using symbolic rules realize it. For each relevant concept of this ontology, agents that extract information according to BWI method achieve the IE task.

The detailed architecture of AGATHE's new extraction cluster is presented in fig. 6. Every *extractor* agent wraps a running TIES for IE, while the *classifier* agent realizes semantic classification only.

An *extractor master* agent is introduced in this cluster in order to ensure system modularity and agent specialization. This agent manages the allocation of IE tasks to *extractor* agents according to a predetermined strategy. Finally, the *storage* agent stores extracted information in a database for future analysis.

C. Information extraction combining agents and wrapper induction

In this section, we detail the three new agents in the new architecture of AGATHE's Extraction Subsystem assuring IE tasks with wrapper induction technique (BWI method):

1) The classifier agent

As defined in previous version of AGATHE, this agent classifies web pages semantically, using symbolic Jess rules and exploiting the domain ontology; if the page belongs to a relevant class of the domain, it sends it to the *extractor master* agent, otherwise it sends it to the *recommendation* agent of its cluster. Moreover, the page's class (ex. Conference, workshop, journal, etc.), with its address as well as other details are sent by this agent to the *storage* agent to be eventually stored in the database.

2) The extractor master agent

Web pages belonging to a specific class of the considered domain might contain information related to different concepts in the domain ontology. While AGATHE has a multi-agent architecture, it appeared coherent and beneficial to distribute the IE load among multiple *extractor* agents according to user-defined strategies. Consequently, it was legitimate to introduce the *extractor master* agent to dispatch IE tasks; first; it receives a classified web page then it resends it toward the *extractor* agents specialized in its class according to the defined distribution strategy.

Fig. 4 illustrates the class based distribution strategy actually adopted in AGATHE. Each *extractor* agent is specialized in extracting information from pages belonging to a specific class in the scientific domain.

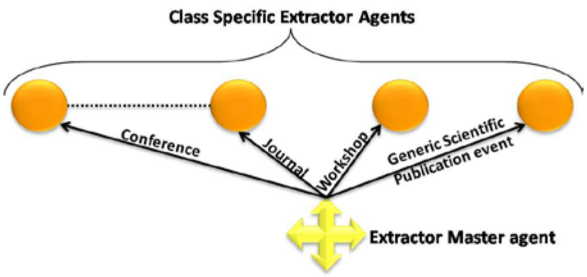
Class Specific Extractor Agents
Conference
Journal
Workshop
Generic Scientific Publication event
Extractor Master agent

Figure 4. Distribution strategy in the extraction subsystem

3) The extractor agent

This agent wraps an adapted version of TIES, the IE system previously presented. It extracts relevant information from web pages by executing TIES, using wrappers produced off-line during a training step and saved in XML files. Choosing a concept-based distribution strategy, a single domain concept is delegated to each *extractor*. Consequently, as soon as an *extractor* agent receives a message, it runs TIES to extract a specific concept from the page specified in the message. Wrapped inside AGATHE, TIES is always executed in "IE" mode; supervised training is carried out independently.

As for all AGATHE's agents, this agent was implemented in JADE platform using two kinds of behaviors; *ExtraTIESProcessor* which is an instance of CyclicBehaviour that receives messages sent by the *extractor master* agent and then passes its content to the *ExtraTIESAnalyze*. The latter is an instance of OneShotBehaviour. It executes an adapted version of TIES, extracts information from the concerned web page, and returns control to *ExtraTIESProcessor*. *Extractor* agent implementation is illustrated in fig. 5.

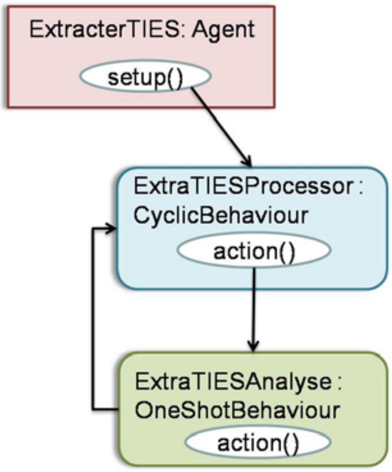
ExtracterTIES: Agent
setup()
ExtraTIESProcessor : CyclicBehaviour
action()
ExtraTIESAnalyse : OneShotBehaviour
action()

Figure 5. Jade Behaviors of the Extractor agent

As presented in fig. 6, system execution goes through the following steps:

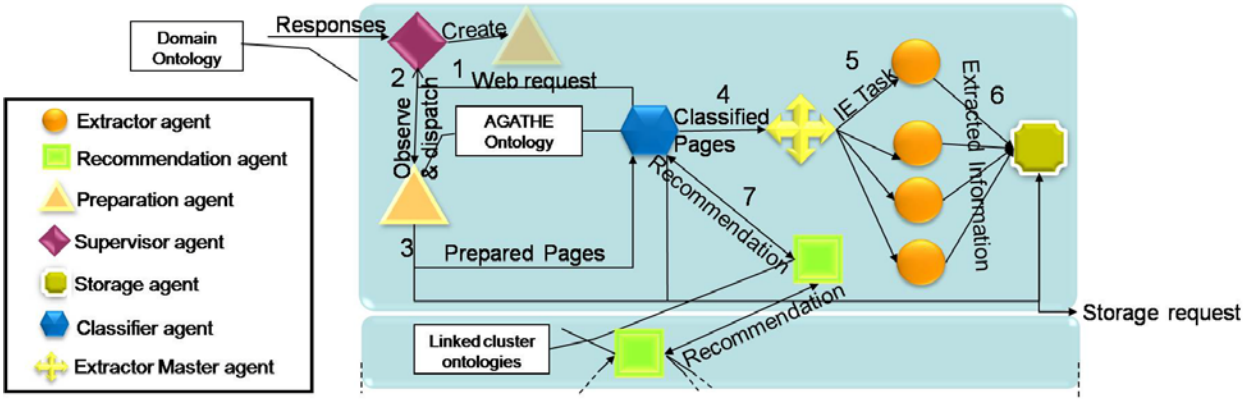

Figure 6. The new Extraction Cluster architecture

1. The *classifier* agent sends a demand for a web search to the *supervisor* agent.
2. The *supervisor* agent forwards the demand towards the search subsystem and then sends the retrieved pages to the *preparation* agent.
3. After filtering and functionally classifying (discarding emails and lists), *preparation* agent sends valid web pages to the *classifier* agent and sends preparation results to the *storage* agent.
4. Web pages belonging to the cluster's specific domain are sent by the *classifier* agent, after classifying them semantically, to the *extractor master* agent.
5. The *extractor master* agent dispatches IE task between multiple *extractor* agents according to a predetermined distribution strategy.
6. Each *extractor* agent realizes the assigned IE task and sends extracted information to the *storage* agent in order to be stored in the database.
7. If the *classifier* agent discovers that the web page doesn't belong to its domain, it forwards it to the *recommendation* agent who decides to which cluster the page must be sent.

V. IMPLEMENTATION DETAILS AND FIRST RESULTS

This new version of AGATHE integrating wrapper induction for IE task is currently under development, and this section presents some implementation details as well as first results obtained.

A. *Implementation details*

The AGATHE architecture is implemented using Eclipse environment in Java, and based on Jade multi-agent platform [20]. The Search Subsystem and the Front Office Subsystem are composed of agents developed in java. The Extraction Subsystem is composed of agents that execute Jess symbolic rules [21]. These agents exploit domain ontology and an internal ontology in order to perform the semantic classification task. For information extraction task, this subsystem deploys agents that wrap an adapted version of TIES. Moreover, the actual AGATHE contains only one extraction cluster without recommendation mechanisms. Finally, the construction and the management of the system's ontologies are realized in the Protégé environment [22], and treatment results are stored in a MySQL relational database system.

B. *First results*

In order to evaluate the new system's performance, we used in our experiments the Pascal test corpus (200 pages containing 2274 annotated fields) without taking into consideration text annotation. Being executed on a single machine supplied with an Intel core 2 Duo (2 GHz) CPU and 3GB of RAM, one page treatment took about (24 seconds) whereas whole corpus treatment took (46 minutes). This treatment covers page filtration and classification in addition to information extraction and storage. Concerning TIES configurations, we adopted the value (4) for the *lookahead* as a compromise between performance and resource consumption.

1) Classification results

According to the first version of AGATHE [1], classification task based on symbolic rules resulted in good precision and recall. For this reason, we maintained the symbolic approach based part for the semantic classification.

Statistical results presented in table II shows that AGATHE had some difficulties in classifying some test corpus' pages (200 calls for workshops) as both classes (workshop and conference) are very close. Anyway, AGATHE was able to classify most corpus pages as workshop as it obtained (86% and 85%) rates for the *precision* and the *recall* respectively.

TABLE II. CLASSIFICATION RESULTS

Number of pages	Classified as conference	Classified as Workshop	Unclassified pages
199	28	169	2

2) IE results

The former version of AGATHE [1] didn't deliver promising results concerning IE task. This task was implemented in symbolic rules, which were a heavy load for the system with low accuracy rate. This was the reason for which we adopted an AIE algorithm in the new version that demonstrated considerably promising results.

After statistical analysis for about 10000 database entries retrieved from treating the test corpus, the average value of *F-measure* for workshop concepts was equal to (70%), which seems comparable to TIES evaluation results presented earlier. Furthermore, for some concepts like *workshoplocation* and *workshopdate*, highest values (more than 85%) were observed for both *precision* and *recall*. Fig. 7 illustrates *F-measure* values obtained during our experimentation concerning different domain concepts. Lowest *F-measure* values were obtained for *conferenceacronym* and *conferencehomepage* concepts. Such low values come from the low frequency of these concepts in the training corpus; as it does not contain enough instances of both concepts so TIES was not trained well to extract them (see table I).

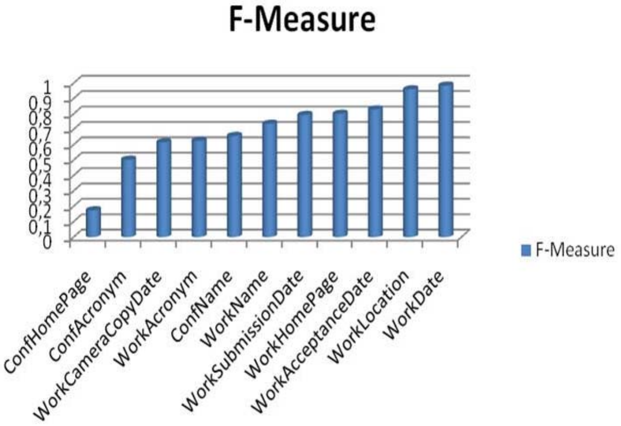

Figure 7. F-measure of extracted concepts in Pascal corpus

3) Illustration

Here we demonstrate how AGATHE works taking as example the call for paper of the conference RCIS2010 presented in fig. 8.

TABLE III. EXTRACTED INFORMATION FROM RCIS' CALL FOR PAPERS

Concept	Value	Start	End
Conferencename	International Conference on RESEARCH CHALLENGES	304	351
Conferenceacronym	RCIS)	376	381
Workshopdate	MAY 19 - 21 , 2010	104	119
Workshophomepage	deadline : November 10 , 2009 http : / / www . farcampus . com / rcis	152	209
Workshoplocation	NICE , FRANCE	121	133
Workshoppapersubmissiondate	deadline : NOVEMBER 10	4488	4509
Workshopnotificationofacceptancedate	NOVEMBER 10 , 2009	4498	4515

First, the page was prepared and then classified as conference that is the correct class of the page. Then, the extractor agent, using TIES wrapped inside it, extracted information from the page. The resulting information, highlighted in fig. 8, is presented in table III.

As workshop and conference classes are so close, they share many concepts like date, location etc. This was the case of last five extracted concepts.

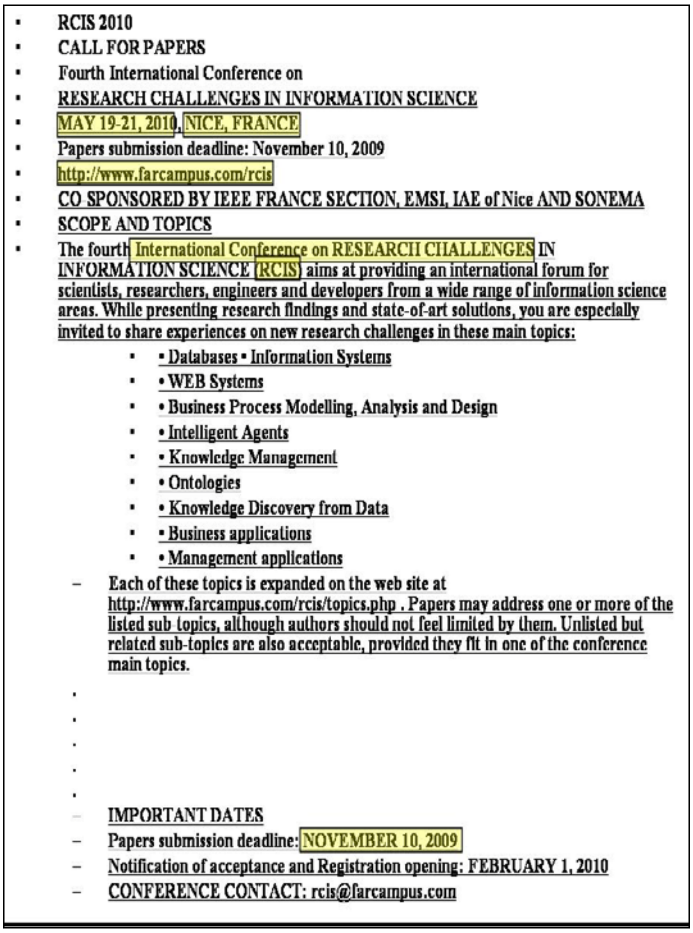
- RCIS 2010
- CALL FOR PAPERS
- Fourth International Conference on
- RESEARCH CHALLENGES IN INFORMATION SCIENCE
- MAY 19-21, 2010, NICE, FRANCE
- Papers submission deadline: November 10, 2009
- http://www.farcampus.com/rcis
- CO SPONSORED BY IEEE FRANCE SECTION, EMSI, IAE of Nice AND SONEMA
- SCOPE AND TOPICS
- The fourth International Conference on RESEARCH CHALLENGES IN INFORMATION SCIENCE (RCIS) aims at providing an international forum for scientists, researchers, engineers and developers from a wide range of information science areas. While presenting research findings and state-of-art solutions, you are especially invited to share experiences on new research challenges in these main topics:
    - • Databases • Information Systems
    - • WEB Systems
    - • Business Process Modelling, Analysis and Design
    - • Intelligent Agents
    - • Knowledge Management
    - • Ontologies
    - • Knowledge Discovery from Data
    - • Business applications
    - • Management applications
- Each of these topics is expanded on the web site at http://www.farcampus.com/rcis/topics.php . Papers may address one or more of the listed sub topics, although authors should not feel limited by them. Unlisted but related sub-topics are also acceptable, provided they fit in one of the conference main topics.
- .
- .
- .
- .
- .
- IMPORTANT DATES
- Papers submission deadline: NOVEMBER 10, 2009
- Notification of acceptance and Registration opening: FEBRUARY 1, 2010
- CONFERENCE CONTACT: rcis@farcampus.com

Figure 8. RCIS call for paper

As TIES used a training corpus to learn how to extract information, similar information contexts are expected in pages to extract or else our system won't be able to extract target fields. This was the case for the concept *workshopnotificationofacceptancedate* that TIES could not extract well. The reason to this error is that three words "and registration opening" were placed after "notification of acceptance" taking the place expected for the acceptance date as it was learned and registered in detectors' patterns. Consequently, TIES did not use the right detectors as it could not find a match for their patterns in the text and so the date was not extracted. This kind of error is explainable as training corpus might not cover all possible cases.

Other sources of confusion might be in the length and the position of field. This was the case for the *conferencename* that was not entirely extracted. Nevertheless, this kind of confusion might be resolved as AGATHE permits user intervention to decide on the correct result.

VI. CONCLUSION

The use of supervised ML techniques in IE, as Boosted Wrapper Induction (BWI), in information gathering on restricted web domains, appears very relevant. Results obtained from this new version of AGATHE, combining agent and wrapper induction in its information extraction task, are very encouraging.

In short-term perspective, we intend first to reduce IE task time, especially by using efficient distribution strategies to dispatch this task. Multiple class specialized extractor agents might be deployed in parallel balancing information extraction load among them. Moreover, as classes belonging to the same domain might share some concepts, extractor agents might be specialized in concepts rather than in classes.

Then, in order to improve the relevance of results, we aim at developing pre and/or post treatments exploiting domain related ontologies. In mid-term, we intend to introduce, in a preprocessing phase, a Part-of-Speech (POS) tagging, in order to improve the IE process by taking into account, always with BWI algorithm, the morphosyntactic structure of the natural language as proposed in [23]. Finally, comparative study is intended in order to evaluate AGATHE considering other state of the art information gathering systems.

VII. ACKNOWLEDGMENT

The authors would like to thank Rinaldo Lima and Fred Freitas of UFCP for their collaboration on AGATHE system, and Alberto Lavelli of IRST for his support on TIES system.

REFERENCES

[1] Espinasse, B., S. Fournier, and F. Freitas, AGATHE: An Agent- and Ontology-Based System for Gathering Information about Restricted Web Domains. International Journal of E-Business Research (IJEBR), 2009. 5(3): p. 14-35.

[2] Siefkes, C. and P. Siniakov, An Overview and Classification of Adaptive Approaches to Information Extraction. Journal on Data Semantics IV, 2005: p. 172-212.

[3] Etzioni, O., M. Banko, S. Soderland, and D.S. Weld, Open information extraction from the web. Commun. ACM, 2008. 51(12): p. 68-74.

[4] Kushmerick, N., Gleaning the Web. IEEE Intelligent Systems, 1999. 14(2): p. 20-22.

[5] Turmo, J., A. Ageno, and N. Catala, Adaptive information extraction. ACM Comput. Surv., 2006. 38(2): p. 4.

[6] Maynard, D., W. Peters, and Y. Li, Metrics for evaluation of ontology-based information extraction, in WWW 2006 Workshop on "Evaluation of Ontologies for the Web" (EON), Edinburgh, Scotland. 2006.

[7] Tang, J., M. Hong, D. Zhang, B. Liang, and J. Li, Information Extraction: Methodologies and Applications, in Emerging Technologies of Text Mining: Techniques and Applications. 2007, Prado and Edilson Ferneda (Ed.), Idea Group Inc., Hershey, USA. p. 1-33.

[8] Muslea, I., S. Minton, and C. Knoblock (1998) STALKER: Learning extraction rules for semistructured Web-based information sources.

[9] Califf, M.E. and R.J. Mooney, Bottom-up relational learning of pattern matching rules for information extraction. J. Mach. Learn. Res., 2003. 4: p. 177-210.

[10] Ciravegna, F., (LP)2: Rule Induction for Information Extraction Using Linguistic Constraints. Technical report. 2003.

[11] Freitag, D. and N. Kushmerick, Boosted Wrapper Induction, in Proceedings of the 17th National Conference on AI and 12th Conference on Innovative Applications of AI. 2000, AAAI Press / The MIT Press.

[12] Finn, A. and N. Kushmerick. Information Extraction by Convergent Boundary Classification. in Proceedings of AAAI-2004 Workshop on Adaptive Text Extraction and Mining. 2004.

[13] Giuliano, C., A. Lavelli, and L. Romano. Simple Information Extraction (SIE): A Portable and Effective IE System. in Proceedings of the EACL-06 Workshop on Adaptive Text Extraction and Mining (ATEM-2004) 2006. Trento, Italy.

[14] Kauchak, D., J. Smarr, and C. Elkan, Sources of Success for Boosted Wrapper Induction. J. Mach. Learn. Res., 2004. 5: p. 499-527.

[15] IST-Dot.Kom. Available from: http://www.dot-kom.org/.

[16] Pascal Challenge. Available from: http://nlp.shef.ac.uk/pascal/.

[17] Ireson, N., F. Ciravegna, M.E. Califf, D. Freitag, N. Kushmerick, and A. Lavelli, Evaluating Machine Learning for Information Extraction, in Proceedings of the 22nd international conference on ML. 2005, ACM: Bonn, Germany.

[18] Freitas, F. and G. Bittencourt. An Ontology-Based Architecture for Cooperative Information Agents. in International Joint Conference on Artificial Intelligence (IJCAI). 2003. Acapulco, Mexico.

[19] Pazienza, M.T., A. Stellato, and M. Vindigni, Combining Ontological Knowledge and Wrapper Induction Techniques into an e-Retail System, in Workshop on ATEM03 held with ECML/PKDD 2003, Cavtat. 2003.

[20] Java Agent DEvelopement Framework. Available from: http://jade.tilab.com/.

[21] Jess, the Rule Engine for the JavaTM Platform Available from: http://www.jessrules.com/.

[22] The Protégé Ontology Editor and Knowledge Acquisition System. Available from: http://protege.stanford.edu/.

[23] Lima, R., B. Espinasse, and F. freitas, Adaptive Information Extraction System based on Wrapper Induction with POS Tagging. To appear in Proceeding of SAC-ACM 2010, Sierre, Switzerland.