Semi-Automatic Population of Ontologies from Text

Tech Report kmi-04-18

David Celjuska and Dr. Maria Vargas-Vera

Semi-Automatic Population of Ontologies from Text

Dávid Čeljuska[1] and Dr. Maria Vargas-Vera[2]

[1] Department of Artificial Intelligence and Cybernetics, Technical University Košice,
Letna 9/A, 04001 Košice, Slovakia
davidmdsk@centrum.sk
[2] KMi - Knowledge Media Institute, The Open University,
Walton Hall, Milton Keynes MK7 66A, United Kingdom
m.vargas-vera@open.ac.uk

Abstract. This paper describes a system for semi-automatic population of ontologies with instances from unstructured text. The system is based on supervised learning and therefore learns extraction rules from annotated text and then applies those rules on newly documents for ontology population. It is based on three componentes: Marmot, a natural language processor; Crystal, a dictionary induction tool; and Badger, an information extraction tool. The important part of the entire cycle is a user who accepts, rejects or modifies newly extracted and suggested instances to be populated. A description of experiments performed with text corpus consisting of 91 documents is given in turn. The results cover the paper and support a presented hypothesis of assigning a rule confidence value to each extraction rule to improve the performance.

Introduction

Ontologies are popular in a number of fields such as knowledge engineering and representation, qualitative modeling, database design, information modeling and integration, object-orientated analysis, information retrieval and extraction, knowledge management, agent systems, and more (Guarino; 1998). In addition to those fields, research analyst companies report on the critical roles of ontologies in areas such as, browsing and searching for e-commerce, and for interoperability for facilitation of knowledge management and configuration (McGuinness; 2002).

However, the problem of their construction and engineering remains not to be completely solved and their development today is more craft than a science. Automated ontology construction tools provide little support to knowledge acquisition. Therefore, the ontology construction process becomes time consuming and this leads to the fact that their wide usage has been limited.

A number of proposals have been published to facilitate ontology engineering (Vargas-Vera et al.; 2001; Craven and Kumilien; 1999; Faure and N'edellec; 1998).

Information Extraction could be considered as a technology that might help an ontology expert during the ontology population and maintenance process.

Information extraction could be seen as the task of pulling predefined entities, objects such as name of visitor, location, date, and so on from texts.

1 Goal of the system and assumptions

Designed system, Ontosophie, in which its framework was motivated by Info-Extractor (Vargas-Vera et al.; 2001) and MnM (Vargas-Vera et al.; 2002) is capable of semi-automatic population of given ontology O with instances. The instances are extracted automatically from natural language text such as plain text or HTML. The task is to identify important entities, slot values $v_1, v_2, \ldots, v_{Ni}$ such as (visitor, date, location, and so on – depending on a class C_i) in a document and thus to construct a vector $V_{ij} = (v_1, \ldots, v_{Ni})$ for each class $C_i \in C_1, C_2, \ldots, C_M$ in the given ontology O. In the next step, it is necessary to determine whether the constructed instance described by V_{ij} vector for class C_i is correct and whether it should be fed into the class or not. This determination is based on the extracted entities and their confidence which will be described later.

Experiments were performed by using KMi's[3] Event ontology O. This consists of events or activities that are defined formally in the ontology as classes $C_i : i \in \{1, 2, \ldots, M\}$. A small part of the ontology is shown in figure 1[4]. Each class/event C_i is defined with set of slots $s_1, s_2, \ldots, s_{Ni}$, which might be instantiated into $v_1, v_2, \ldots, v_{Ni}$. Type of each slot is in default *String* which gives high flexibility in terms that all integers, floats, dates, strings, list of names and so on could be expressed in a string form.

The following part shows one class definition from the event topology in order to explain its structure:

```
Class Event: Conferring-an-Award
Description: Class of an event describing an event of awarding someone
Slots:
has-duration (when or how long the event took)
has-location (a place where it took place)
recipient-agents (the agents who received the award)
has-awarding-body (an organization, donor)
has-award-rationale (for what the award is)
object-acted-on (award, name of the award or amount of money)
```

One might notice that this particular class is named *Conferring-an-Award* and it describes any even talking about awarding someone an award for some reason. The slots such as *has-duration*, *has-location*, and so on are attributes of the class and give detailed information about one specific event/instance which the system is intended to construct. The task of the system is therefore to identify those attributes $v_1, \ldots, v_{Ni}$, entities in a document and then to construct an instance and fed it into appropriate class C_i within the ontology O.

[3] Knowledge Media Institute, The Open University, United Kingdom

[4] classes that this paper refers to are highlighted

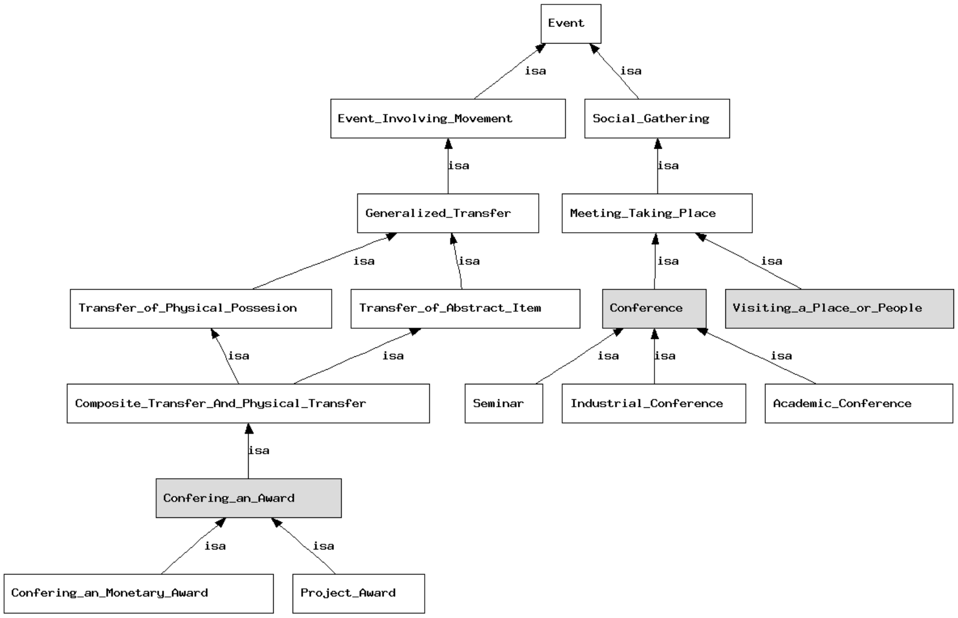


Fig. 1. A Part of Event Ontology

2 System's framework

The system's framework consists of the following parts (figure 2):

- Annotation/Mark-up
- Learning
- Extraction
- Population

Each of these steps will be explained in turn.

3 Annotation

The activity of semantic tagging refers to the activity of annotating text articles (written in natural language in plain text or HTML) with a set of tags defined in the ontology.

Each slot that occur within any class of the ontology is assigned a unique XML tag. Thus mark-up step is ontology driven. Once the user identified desired class for a displayed document from the ontology he is offered with relevant tags only. An annotated article might then look as follows:

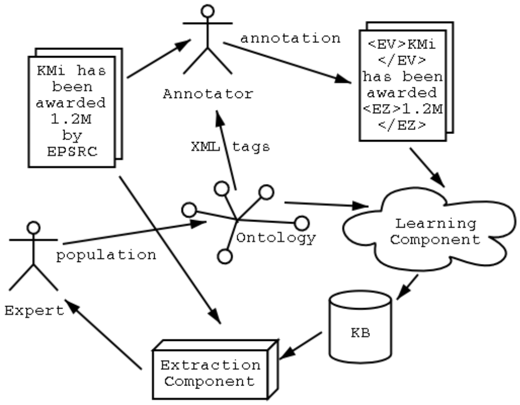


Fig. 2. The framework of the system

> **<EV>KMi</EV>** has been awarded **<EZ>L1.2M</EZ>** by the **<EX>UK's Engineering and Physical Sciences Research Council</EX>** to carry out **<EY>research in the application of knowledge technologies to support knowledge creation and sharing in organizations</EY>**.

One might notice that the tag *EV* refers to *recipient-agent*, an agent who was given an award, *EX* to *has-awarding-body*, a name of the organization which gave, sponsored the award, *EZ* to *object-acted-on*, the award itself and *EY* to *has-award-rationale*, the reason of the award or for what the award was.

Once a set of documents is annotated with XML tags and all articles are stored, the following learning phase may begin.

4 Learning

Thus the system is based on supervised learning, the training set of documents is required. The learning set in this context means a set of annotated articles (section 3). Learning consists of two steps:

- Natural language processing
- Learning extraction rules

Each of the steps will be described in detail as follows.

4.1 Natural language processing

Natural language analysis is extremely crucial and is very often under-estimated.

Ontosophie, as most information extraction systems, uses shallow parsing to recognize syntactic constructs without generating a complete parse tree for each sentence. Such shallow parsing has the advantages of higher speed and robustness. High speed is necessary to apply the information extraction to a large volume of documents. The robustness achieved by using a shallow parsing is essential to deal with unstructured texts. In particular, Ontosophie uses Marmot[5], a natural language processing system.

Marmot accepts ASCII files and produces an intermediate level of text analysis that is useful for information extraction applications. Sentences are separated and segmented into noun phrases, verb phrases and other high-level constituents.

After each document was annotated and pre-processed with the Natural Language Processing tool, the set of documents enters the Learning phase itself.

4.2 Generating extraction rules

Learning extraction rules from an annotated set of documents is a task of generating a set[6] of extraction rules. Ontosophie in this phase uses Crystal, a dictionary induction system. Crystal[7] (Soderland at al.; 1995) is a conceptual dictionary induction tool, which derives a dictionary of concept nodes, extraction rules, from a training corpus. Crystal is based on specific-to-general algorithm and its purpose is to learn extraction rules – concept node definitions. For illustration, an extraction rule might be understood as following: *conferring-an-award: <VB-P "been awarded"> <OBJ1 any> <PP "by" has-awarding-body> Coverage: 5 Error: 1*[8] The rules purpose is to extract *conferring-an-award*, which refers to name of a class from the ontology O (figure1). This concept node, extraction rule, is defined to extract *has-awarding-body*, name of a donor or sponsor of some award. The rule fires only in case all the constraints are satisfied. This, in particular, means that the entity *conferring-an-award* is extracted from any sentence or its part only in case it consists of *"has been awarded"* as passive verb (VB-P), an object (OBJ1) that might be anything and it contains a prepositional phrase (PP), which starts with preposition *"by"*. When those constraints are satisfied the rule fires, meaning the prepositional phase (PP) is extracted as *has-awarding-body.* For example, from the sentence: *"KMi has been awarded L1.2M by the UK's Engineering and Physical Sciences Research Council to carry out research in . . . "*[9] it will extract *"by the UK's Engineering and Physical Sciences Research Council"* as the particular value of the slot *has-awarding-body.*

In addition to that, Crystal gives two values – *Coverage* and *Error*. In this particular example, the rule covered five instances (one incorrectly) in the corpus in which the rule was generated from. Which gives some-what feel of rule's precision[10].

[5] Marmot was developed at University of Massachusetts, MA, USA

[6] also called dictionary

[7] Crystal was developed at University of Massachusetts, MA, USA

[8] Due to sake of space the extraction rule is shown in a short form. It is much more complex in reality, but this form is enough to explain the most important parts.

[9] this is one of the sentence from the annotated article described in section 3

[10] $P = (5 - 1)/5$

Getting extraction rules by using Crystal is not sufficient as one rule might have higher confidence than another. Thus, computing rule confidence becomes essential.

4.3 Assigning rule confidence values to extracted rules

In most cases having a precise and correct ontology rather than having it overpopulated with incorrect instances is more important. Therefore, in the area of fully-automated ontology population more pressure is applied on precision rather than recall[11]. On the other hand, when dealing with semi-automatic approach, it is often required to have high recall also at a cost of lower precision. In this case users prefer to have higher control over the process and be offered with multiple choices from which they can pick the desired one.

From what was said, the optimal is to keep high recall while in default, automatically pre-select those options that are believed to be precise enough.

In order to achieve this task Ontosophie attaches a rule confidence value to each rule[12]. The rule confidence tells how sure the system is about the thruthness of a particular rule.

Experimentation showed, that some extraction rules that were learned by Crystal are very weak and therefore firing too often, while other rules might be overly specific. In addition, previous experiments (Riloff; 1996) showed that precision improves if those rules are manually removed. However, our approach is to take an automatic control over this. Thus, those rules need to be either eliminated or given low rule confidence value. The extraction rule confidence tells, how sure the system is about its quality in comparison to other rules in the dictionary.

Ontosophie is equipped with two ways of computing the rule confidence value.

The first and most simple method uses *Coverage* and *Error* values that are automatically provided for each rule by Crystal (section 4.2). In this case the rule confidence is computed as:

$$C = \frac{c}{n} = \frac{Coverage - Error}{Coverage} \tag{1}$$

Where c is the number of times the rule is fired correctly and n is the number of times the rule is fired in total. *Coverage* tells how many instances the particular rule covers, or in other words, how many times the rule is fired on the entire training set and *Error* tells how many times it is fired incorrectly.

However, (1) does not distinguish between, for example $C_2 = (2 - 0)/2$ and $C_{10} = (10)/10$, because $C_2 = C_{10} = 1.0$. At this point, one might argue that C_{10} is more accurate and has higher support, because in this case the rule fired ten times out of ten correctly, while the other one only fired two out of two. This is

[11] as a reminder: precision $P = c/n$ and recall $R = c/m$, where c is number of correctly extracted entities, n is the total number of extracted entities and m the total number of entities that should be extracted

[12] None of the mentioned systems including MnM and Info-Extractor has this feature.

why Ontosophie was designed to take this fact into consideration. In particular it uses Laplace Expected Error Estimate (Clark and Boswell; 1991) which is defined as $1 - LaplaceAccuracy$, where:

$$LaplaceAccuracy = \frac{n_c + 1}{n_{tot} + k} \tag{2}$$

where:[13]

- n_c is the number of examples in the predicted class covered by the rule
- n_{tot} is the total number of examples covered by the rule
- k is the number of classes in the domain

Implementing the equation 2 to the valuation of confidence is then:

$$C = \frac{c + 1}{n + 2} \tag{3}$$

Where $k = 2$ because it deals with two classes for each rule. One, the rule fires and two, the rule does not fire. When $k = 2$ a posteriori probability is set to $1/2 = 0.5$ [14]. Meaning, that if $C = 0.5$ the rule fires correctly as often as it does incorrectly. This is the state when nothing serious can be said about the rule and thus all rules with $C \leq 0.5$ should be eliminated.

The other method is more sophisticated and it is based on different mathematical model. In this case the rule confidence is computed independently on *Covers* and *Error* values provided by Crystal.

In this case the confidence number for each rule is computed by the k-Fold Cross validation methodology (Mitchell; 1997) on the training set. At each run a new set/dictionary of extraction rules is generated by Crystal. The algorithm 1 outlines the methodology that Ontosophie uses. The algorithm computes for each rule r_i how many times it fired correctly c_{r_i}, how many times it fired in total n_{r_i}, performs merging of identical rules and assigns x_{r_i} to each rule that tells how many times the rule was merged.

If two rules r_i and r_j generated from two different runs are identical, regarding their constraints, they are merged to form one new rule r_{new} which is identical to the r_i and r_j while the number of times the rule r_{new} is fired correctly $c_{r_{new}} = c_{r_i} + c_{r_j}$ and number of times it is fired in total $n_{r_{new}} = n_{r_i} + n_{r_j}$. After the whole process is then:

$$c_{r_{new}} = \sum_{\forall i,j; i \neq j : constrains(r_i) = constrain(r_j)} c_i + c_j$$

$$n_{r_{new}} = \sum_{\forall i,j; i \neq j : constrains(r_i) = constrain(r_j)} n_i + n_j$$

[13] The presented Laplace Error Estimate is borrowed from Classification, that is why the particular variables are defined as they are.

[14] one might note that $(K + 1)/(2K + 2) = 0.5 : \forall\ K \in \mathcal{R} \rightarrow$ if $c = 2n$ then $C = 0.5$

Algorithm 1 Computation of rule confidence by k-Fold-Cross in Ontosophie

```
$N \leftarrow$ number of classes
$S_{ij} \leftarrow$ is $j$-th fold of $i$-th class, $1 \leq i \leq M$ and $1 \leq j \leq k$
$S_i \leftarrow S_{11} \cup S_{12} \ldots \cup S_{1k}$ {is set of documents of $i$-th class}
$S \leftarrow S_1 \cup S_2 \cup \ldots \cup S_N$ {is entire training set}
$W \leftarrow \emptyset$ {the final set of rules with rule confidence computed for each of them}
for all $j$ such that $1 \leq j \leq k$ do
  $T = V = \emptyset$ {$T$ is a training set and $V$ is a validation set}
  for all $i$ such that $1 \leq i \leq M$ do
    $T \leftarrow T \cup S_i - S_{ij}$ {training set}
    $V \leftarrow V \cup S_{ij}$ {validation set}
  end for
  $R \leftarrow \mathit{generateExtractionRules}(T)$ {generates a set of rules by running Crystal for set $T$}
  $R \leftarrow \mathit{setXtoZero}(R)$ {sets $x$, number of time the rule was merged, to zero for each rule in the set $R$}
  $R_e \leftarrow \mathit{evaluate}(R, V)$ {$R_e$ is set of evaluated $R$ rules with $V$}
  $W \leftarrow W \cup R_e$
end for
while $\exists\ i, j, i \neq j; r_i, r_j \in W : \mathit{constrains}(r_i) = \mathit{constrains}(r_j)$ do
  $r_{new} \leftarrow \mathit{merge}(r_i, r_j)$ {$\mathit{constrains}(r_{new}) \leftarrow \mathit{constrains}(r_i) = \mathit{constrains}(r_j)$}
  $c_{r_{new}} \leftarrow c_{r_i} + c_{r_j}$ {number of times it fired correctly (refers to $c$)}
  $n_{r_{new}} \leftarrow n_{r_i} + n_{r_j}$ {number of times it fired in total (refers to $n$)}
  $x_{r_{new}} \leftarrow x_{r_i} + x_{r_j} + 1$ {counting number of times the rule was merged}
  $W \leftarrow W - \{r_i\} - \{r_j\} + \{r_{new}\}$
end while
```

The evaluation of a rule (algorithm 1) is only one aspect that has not been covered yet. The evaluation is always performed on the validation set, as it is clear from the algorithm. At each run after all the rules have been generated by Crystal, Ontosophie enters evaluation state which is based on the extraction. All rules that were responsible for correctly extracting an entity are then awarded $c_i \leftarrow c_i + 1$. Certainly, n_i is incremented $n_i \leftarrow n_i + 1$ for all rules that were active during the extraction. The tough phase is to recognize whether an extracted entity is correct or not. However this is beyond the scope of this paper – we advice you to have a look at (Čeljuska; 2004) for more details.

5 Extraction and ontology population

Once all the extraction rules are learnt and assigned a rule confidence value, the system is ready for extraction.

The task of this phase is to extract appropriate entities from a document[15] and feed a newly created instance into given Ontology O. The document is preprocessed with Marmot (similarly as described in section 4.1) prior to extraction itself.

The extraction is run class by class. Firstly, a set of extraction rules for only one specific class from the ontology is taken and only those rules are used for the extraction. The step is then repeated for all the classes within the ontology and thus for each class the system gets a couple of entities that correspond to slots from the ontology. Three different outcomes might be observed:

1. None of the entity was extracted and the document then remains unclassified
2. Only entities of one class within the ontology were extracted. It's clear that the document can only belong to this class.
3. Entities from more than one class were extracted. The decision has to be undertaken to determine which classes the instances[16] should be linked to.

Ontosophie is a semi-automatic system and thus in order to give a user a large volume of control without the need of too much interaction, the following has been implemented in Ontosophie.

The user is provided with all the extracted possibilities while automatically pre-selecting those that are believed to be strongly accurate. The figure 3 shows a part of original text[17] and a window dialog with suggestions for ontology population. To give a user control over automatic pre-selection, two threshold numbers are provided for pruning.

However, before the pruning is explained the following description of extraction and slot/class confidence value computation is given.

For the information extraction a third component called Badger[18] was also integrated into the system. Badger makes the instantiation of templates. The

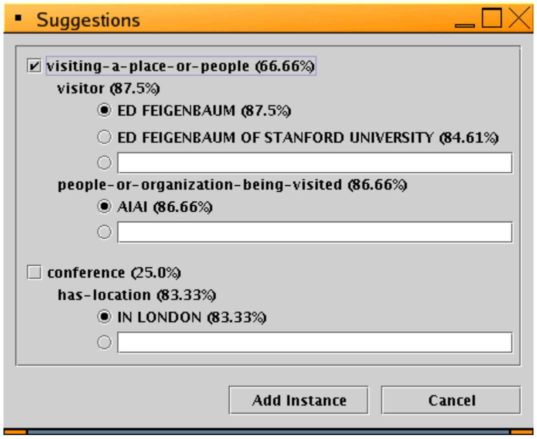
[15] not yet processed nor annotated document

[16] an instance consists of slots and its values (extracted entities)

[17] automatically recognized entities were printed in **bold**.

[18] Badgar was developed at the University of Massachusetts, USA

Ed Feigenbaum Visits **AIAI**
Wednesday, 18th July 2001
Ed Feigenbaum of Stanford University visited **AIAI** on 2nd July 2001 to hear about the knowledge-based systems and applied AI work of the Institute. He heard about the plans to form CISA on 1st August 2001...He is currently working with the European Office of Aerospace Research and. Development in **London**, part of the US Air Force Office of Scientific.



Suggestions

☑ visiting-a-place-or-people (66.66%)
visitor (87.5%)
● ED FEIGENBAUM (87.5%)
○ ED FEIGENBAUM OF STANFORD UNIVERSITY (84.61%)
○
people-or-organization-being-visited (86.66%)
● AIAI (86.66%)
○

□ conference (25.0%)
has-location (83.33%)
● IN LONDON (83.33%)
○

Fig. 3. A part of original text and a dialog with extracted entities

main task of Badger is to take each sentence from the document and see if any of the learnt rules can be applied (section 4.2). If no extraction rule applies to a sentence, then no information is being extracted - irrelevant text is processed very quickly.

Badger in addition to extracted entities gives a list of extraction rule Id's that were responsible for the extraction of a particular entity. This way Ontosophie can pull confidence values C_i for each of the fired rule from the dictionary and performs post-computing and pruning.

It might happen that Crystal extracts more than one value for a given slot name. This is the collision that has to be solved. Therefore, extraction phase might lead to the following problems that have to be undertaken:

- The same piece of information, an entity was extracted with more than one rule - value collision
- More than one value was extracted for a given slot - slot collision
- Entities from different classes where extracted - class collision

The following section will describe solutions given by Ontosophie.

5.1 Solving collisions

In Ontosophie, not only are rules assigned confidence values, but also extracted entities, slots and classes.

If one piece, an entity is extracted by only one rule, then the value confidence C_{value} associated to that piece of information is equal to the confidence of the rule that extracted it $C_{value} = C$. Where rule confidence C is computed by either (3) or (1).

However, if one entity is extracted with more than one rule, then C_{value} is computed as the maximum overall rule confidences of rules that fired it.

$$C_{value} = \max_{\forall i : r_i \text{ is in collision}} C_i \tag{4}$$

The same applies for the slot confidence C_{slot}. If one value was extracted for a given slot (i.e. *visitor = "Ed Feigenbaum"*) then $C_{slot} = C_{value}$. However, if more then one value was extracted for a slot *visitor = "Ed Feigenbaum" and visitor = "Ed Feigenbaum of Stanford University"*, then only the value with its highest confidence is considered. Thus, $C_{slot} = \max_{\forall i : C_{value,i} \text{ is in collision}} C_{value,i}$. The highest scored value/entity for a given slot is then pre-selected and values/entities are ordered by their confidence (figure 3).

It might happen, that the system extracts some entities from one class and some entities from another class (*visiting-a-place-or-people* and *conference* - figure 3). It is important to determine which classes the new instances should be fed into. For this purpose, the class confidence value C_{class} is assigned to each class. Although there is more than one way of computing class confidence C_{class} we used the following equation:

$$C_{class} = \frac{\sum_{\forall \text{ extracted slots for the given class}} C_{slot,i}}{m_{class}} \tag{5}$$

Where m_{class} is a number of different slots the system is able to fill in with extracted entities for a class *class*.

5.2 Pruning

There are two different threshold values. One is for pruning slots and one for classes. When in the extraction phase some slot is assigned a slot confidence value $C_{slot} < Threshold_{slot}$ then this slot is not pre-selected and also do not play any role in the phase of computing class confidence value (section 5.1). Otherwise it is pre-selected.

The second threshold value $Threshold_{class}$ is used in case of classification. Classes that have confidence $C_{class} < Threshold_{class}$ are not pre-selected.

Threshold values are very helpful to speed up the process of rejecting/accepting. In case a user is offered only with trusted and confident pre-selections the high volume of interaction is avoided and this goal of Ontosophie is achieved.

After the extraction process is finished, a users interaction is required to take the final decision about the extracted instances. The user has the ability to reselect pre-selected options or completely reject to populate the ontology with any instance. However as it was stated above, the goal of the system is to automatically fill as many slots as possible while only pre-select those values/slots/classes that are most likely to be correct based on the threshold values.

6 Validation of the system and experiments

6.1 Description of the set

For the experiments the ontology described in section 1 was used. Although, it contains 41 classes, only three of them were used for the experimentation due to low number of different kind of documents talking about different events.

Particular short text articles, similar to the example given in section 3, were gathered from five different archives. Thus they were not all in common format they were normalized by using a little Java program into one uniform form.

All of the articles were annotated, as described in section 3, and every document was classified into one and only one of the mentioned class.

Table 1. Statistical information for each class and its slots. Pos. – number of positive instances per slots; Tot. pos. – number of positive instances per class; Total – number of instance including positive and negative; Docs – number of documents per class

Class	Slot name	#Pos.	#Tot. pos.	# Total	# Docs
conference	has-duration	31	205	561	27
	has-location	38			
	main-agent	69			
	meeting-attendees	50			
	meeting-organizer	17			
conferring-an-award	has-duration	10	206	517	29
	has-location	3			
	recipient-agents	79			
	has-awarding-body	30			
	has-award-rationale	30			
	object-acted-on	54			
visiting-a-place-or-*	has-duration	35	272	707	35
	has-location	6			
	visitor	132			
	people-or-org*	99			

The table 1 shows a number of documents for each of the class (*“Docs”*). In addition, it gives number of annotated entities (positive instances - *“Pos.”*), number of positive instance (*“Tot. pos.”*) in total for a given class and total number of instances for each class (*“Total”*). Slots that have not been annotated within the entire dataset are not listed.

6.2 Validation of the system

The goal of the validation of the system was to give answers to the following questions:

- How using rule confidences effects precision and recall.

- Which of the two ways of computing the rule confidence (section 4.3) is better.
- How elimination of rules effects the precision and recall.

When the rule confidence is being computed by k-Fold-Cross methodology, as it was stated it section 4.3, at each run a new set of rules is generated by Crystal. Then the rules that are identical are merged and x_i which tells how many time a rule r_i was merged, is computed (algorithm 1). It is believed, that a rule which was generated from more than one run is more likely to do good in the entire set and not just within the part it was generated from. Thus, it is believed that removing all rules $r_i : \forall i; x_i < Merge$ might result in better quality rule dictionary (set). The parameter $Merge$ controls which rules will be kept in the dictionary and which will be removed. For example if $Merge = 1$ then the system will remove all the rules that were not merged at least once. Or by other words, rules that were only generated from one run, fold.

Four different experiments were run. For the validation of each of the experiment the 5-fold-cross[19] validation methodology was used. More over, each experiment was repeated five times to get better statistical results and in case of experiments where k-Fold-Cross was used to compute rule confidence the dataset was randomly split each time into k folds.

The following experiments were run:

1. *"No confidence"* – without using any rule confidence value. Thus no pruning was used. The rules were treated as equal without any preferences and all generated rules were used.
2. *"Simple"* – the first, simple method for computation of rule confidence was used - equation (1) with Laplace error estimate: $C = \frac{c+1}{n+2} = \frac{Coverage-Error+1}{Coverage+2}$
3. *"k-Fold"* – the second, k-Fold-Cross validation method was used for computing the rule confidence. The k was set to 5 so in particular 5-fold-cross was used. No rule was eliminated thus $Merge = 0$ and all the rules generated from each run/fold were used.
4. *"Elimination"* – similarly to *"k-Fold"*, 5-fold-cross was used to compute rule confidence. However, this time rules that did not show up from at least 3 folds were removed: $Merge = 2$.

Computation of class confidence was done by (5) for the experiments from 2 to 4. In addition threshold values in case of experiments 2 – 4 were set as follows: $Threshold_{class} = 0.3$ and $Threshold_{slot} = 0.7$.

The table 2 shows precision (P) and recall (R) for each of the experiment and for each of the class separately. The presented values shows the minimum, maximum and the average values observed throughout the five trials.

One might notice from the table, that the variability[20] almost crosses 10% in case of *"Elimination"*, which implies that the computed average values is not statistically very reliable. The figures 4 and 5 gives better picture of the results.

[19] 5-fold-cross validation is not a standard. Most of the time 10-fold-cross is used. However, to save processing time the 5-fold-cross was used instead

[20] difference between an average, minimum and maximum

Table 2. Comparison of different experiments

		No confid.		Simple		k-Fold		Elimination	
		P (%)	R (%)	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
Conference	Max	80.25	10.86	100.00	7.23	-	0.00	-	0.00
	Min	51.78	7.11	100.00	0.72	-	0.00	-	0.00
	Avg	67.11	8.75	100.00	4.38	-	0.00	-	0.00
Award	Max	85.41	11.00	93.33	8.71	100.00	5.72	100.00	1.76
	Min	74.85	7.10	76.28	6.24	81.25	3.26	50.00	0.00
	Avg	81.46	9.19	82.49	7.82	92.47	4.58	66.67	0.57
Visiting	Max	72.36	30.25	81.02	17.51	90.16	14.71	96.66	11.99
	Min	65.83	25.74	73.34	15.95	81.38	12.13	86.00	7.83
	Avg	70.32	28.11	77.57	16.78	84.97	13.16	90.77	9.87
Total	Max	72.17	16.96	80.81	10.64	88.33	6.76	94.66	5.28
	Min	65.86	15.40	76.10	9.66	84.38	6.31	81.39	3.30
	Avg	69.45	16.10	78.38	10.19	85.62	6.53	89.22	4.09

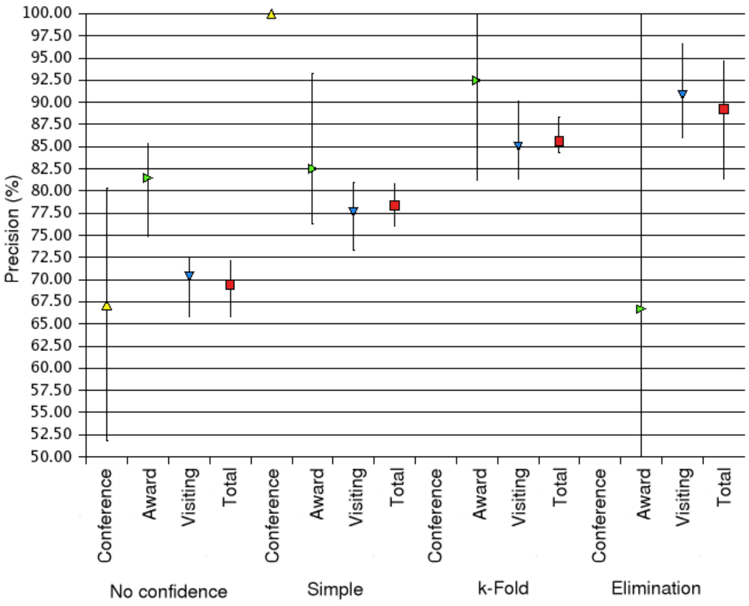


Fig. 4. Precision for each method - per class and in total

It can be observed from the table 2, that there is significant change in precision between cases when the rule confidence is taken into consideration and not. The recall however goes rapidly down and variability increases. Only looking at total precision it might seem that *"Elimination"* is the best choice with its 89.22% average. However, as one can see, the recall is extremely low – for the class *conference* no entity was extracted. Even more in this case the precision at *conferring-an-award* is, comparable to the total, very low. A deeper analysis of

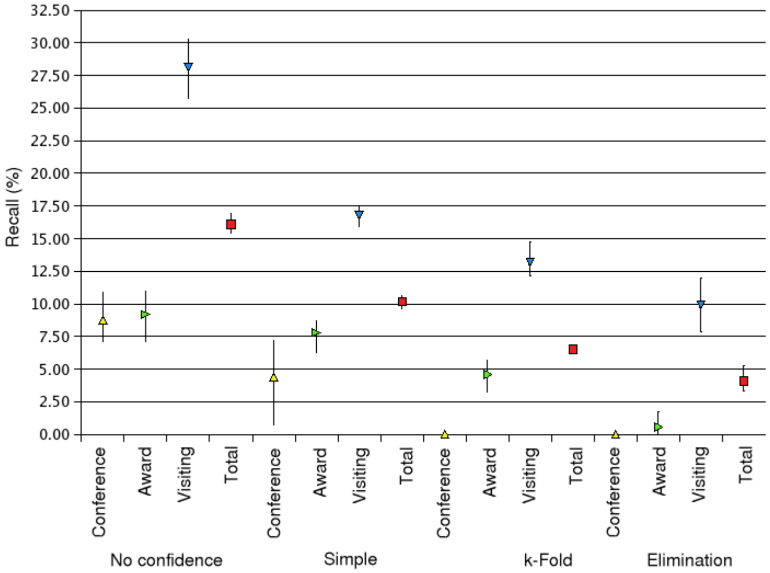


Fig. 5. Recall for each method - per class and in total

this particular case showed that throughout all five trials the slot *recipient-agent* was extracted only two times correctly out of four and the slot *object-acted-on* three out of five. From all of the possible 545 positive instances[21] only 5 were extracted correctly from 9 tries. This is why this particular result does not significantly affect the total average precision of the experiment. For example in case of *visiting-a-place-or-people* the system for the total five trials extracted 132 times correctly out of 147 tries from possible 1326 positive instances.

"k-Fold-Cross" obtains a lower average of total precision 85.62% while recall is a little higher. At each class it went well, besides *conference*. It might be considered as one of the better options from all the experiments.

It can be concluded from the experiments that using a rule confidence is a big plus. It also seems that k-Fold-Cross methodology is a better choice to the simple method if in search for high precision and not so recall depended output. Elimination of rules in case of *"Elimination"* needs to be taken with care. In the experiment the rules that were not generated from at least three different runs out of five were strictly removed. It was observed from examples that from 79 rules it resulted into 24 after the elimination. This is quite a drastic pruning. The figure 5 shows recall for each of the experiments and thus gives better picture on how the recall drops by increasing the precision.

It is also important to note that the dataset (table 1) was not very big. Thus the absolute number of positive instances is not very high. Learning on such

[21] $num\_trials(num\_positive\_for\_recipient\_agent + num\_positive\_for\_object\_acted\_on) = 5(79 + 30) = 545$

a small set is always leading to more specific rules or overly generalized and causing variability to go high.

7 Conclusion and future work

Ontology construction process is very time consuming and as a consequence, systems for semi-automatic acquisition of ontologies from text are being developed.

This paper described designed system Ontosophie which is based on three components: a Natural Language Processing component called Marmot, a dictionary induction tool named Crystal and an Information Extraction component called Badger. All three of these are from the University of Massachusetts, MA, USA.

In the area of semi-automatic population it is important to have a system that gives a user the control over the process while automatically offering only the most trusted and believed to be correct suggestions for the ontology population. Ontosophie is equipped with this feature in terms that at the extraction phase it always performs the full extraction while pre-selecting only those suggestions that are considered to be correct. This is done by applying a pruning based on threshold parameters set by a user. In order to evaluate an extracted entity, two different designed methods for computing rule confidence were introduced. The experiments conducted using those methods showed that using the rule confidence might increase the precision by around 15% depending on different models and parameters. In addition, it was observed that using the k-Fold-Cross methodology for computation seems to be a better choice to the simple method of taking *Coverage* and *Error* values computed by the learning component Crystal.

The system was also tested with a third party user who did not have any prior information about the system's framework. Although the user reported that the system was fairly straight forward to use once it was set up, he did mention that it was quite difficult to determine the right class and its extracted slot values, just being based on the extraction dialog. This fact was taken into consideration and for the next generation of Ontosophie it is suggested to perform text highlighting of the extracted information in the original document, which the extraction is run from. This way a user could, by clicking on each of the suggestions, see the extracted entities within a context and also determine the right values and desired classes much more quickly and precisely. In addition, the next generation should provide a more superior post-processing tool that could also include the entity type validation. This could be done by comparing the type of the slot and the type of the extracted information as also suggested by (Vargas-Vera and Čeljuska; 2003).

References

CLARK, P. – BOSWELL, R. 1991. *Rule induction with CN2: Some recent improvements.* Proceeding Fifth European Working Session on Learning, pages 151 – 163, Springer, Berlin.

CRAVEN, M. – KUMILIEN, J. 1999. *Constructing Biological Knowledge Bases by Extracting Information from Text Sources.* Proceedings of The 7th International Conference on Intelligent Systems for Molecular Biology (ISMB-99).

ČELJUSKA, D. 2004. *Semi-automatic Construction of Ontologies from Text.* Master's Thesis, Department of Artificial Intelligence and Cybernetics, Technical University Košice.

FAURE, D. – N'EDELLEC C. 1998. *ASIUM: Learning sub-categorization frames and restrictions of selection.* 10th Conference on Machine Learning (ECML 98) – Workshop on Text Mining, Chemnitz, Germany.

GUARINO, N. 1998. *Formal Ontology and Information Systems.* Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98, Trento, Italy, pages 3 – 15. IOS Press.

McGUINNESS, D. L. 2002. *Ontologies Come of Age.* Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press.

MITCHELL, T. 1997. *Machine Learning.* McGraw-Hill, ISBN 0070428077.

RILOFF, E. 1996b. *Automatically Generating Extracting Patterns from Untagged Text.* Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), pp 1044-1049.

SODERLAND, S. – FISHER, D. – ASELTINE, J. – LEHNERT, W. 1995. *CRYSTAL: Inducing a Conceptual Dictionary.* Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pages 1314 – 1319.

VARGAS-VERA, M. – DOMINGUE, J. – KALFOGLOU, Y. – MOTTA, E. – BUCKINGHAM SHUM, S. 2001. *Template-Driven Information Extraction for Populating Ontologies.* In proceedings of the Workshop Ontology Learning IJCAI-2001, Seattle, USA.

VARGAS-VERA, M. – ČELJUSKA, D. 2003. *Event Recognition using Information Extraction Techniques.* KMI-TR-134, Knowledge Media Institute, The Open University.

VARGAS-VERA, M. – MOTTA, E. – DOMINGUE, J. – LANZONI, M. – STUTT, A. – CIRAVEGNA, F. 2002. *MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Mark-up.* The 13th International Conference on Knowledge Engineering and Management (EKAW 2002), Lecture Notes in Computer Science 2473, ed Gomez-Perez, A., Springer Verlag, 2002, 379-391, ISBN 3-540-44268-5.

MARMOT User Guide. Natural Language Processing Laboratory, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, USA.

Task Domain Specification and User Guide for Badger and Crystal. Natural Language Processing Laboratory, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, USA.