Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a umelej inteligencie

Semiautomatická Konštrukcia Ontológií z Textov

Diplomová práca

Študijný odbor: Umelá Inteligencia

Vedúci diplomovej práce:

Ing. Ján Paralič, Phd.

Diplomant:

Dávid Čeljuska

Konzultant diplomovej práce:

Dr. Maria Vargas-Vera

Košice 2004

Technical University Košice

Faculty of Electrical Engineering and Informatics

Department of Cybernetics and Artificial Intelligence

Semi-Automatic Construction of Ontologies from Text

Master's Thesis

Study branch: Artificial Intelligence

Supervisor:

Ing. Ján Paralič, Phd.

Author:

Dávid Čeljuska

Consultant:

Dr. Maria Vargas-Vera

Košice 2004

Čestné vyhlásenie

Vyhlasujem, že som diplomovú prácu vypracoval(a) samostatne s využitím uvedenej odbornej literatúry.

Košice 9. 5. 2004 . . . .

Vlastnoručný podpis

zadanie diplomovky

Acknowledgement

I am glad that I had an opportunity to work with such wonderful people as Dr. Maria Vargas-Vera and others from KMi (Knowledge Media Institute at The Open University, United Kingdom). My big thanks goes out to Maria, who has supported and found time for me when I needed questions answered. Maria has been more like a friend than a consultant and I am thankful for her reviews, great suggestions, outstanding advice and the tremendous amount of hours spent on ICQ.

I would also like to thank my supervisor Ing. Jan Paralic, Phd. without whom I would not have come across this opportunity. I am very grateful for his support both on professional and private level. I appreciate the freedom he has given me and the environment that he created for me.

My thanks also go out to Peter Matthews. Not only has Pete been providing me with excellent black tea from York, which has been keeping me awake for the last couple of weeks during my hectic schedule, but he has also found time to review the document and make grammatical corrections.

And last but not least, thanks to my always supportive sister, parents and friends. Sorry for putting you all on one side for a while – will try to make it up to you in the near future.

Názov práce:	Semiautomatická Konštrukcia Ontológií z Textov
Pracovisko:	Katedra kybernetiky a umelej inteligencie, FEI TU v Košiciach
Autor:	Dávid Čeljuska
Vedúci DP:	Ing. Ján Paralič, Phd.
Konzultant DP:	Dr. Maria Vargas-Vera KMi – Knowledge Media Institute, The Open University, United Kingdom
Dátum:	9. 5. 2004
Kľúčové slová:	systém pre semiautomatické dopĺňanie ontológií o nové inštancie, konštrukcia ontológií, semiautomatické dopĺňanie ontológií o nové inštancie, inštancia, text, spoľahlivosť pravidiel, výpočet spoľahlivosti pravidiel, prirodzený jazyk, extrakcia informácií, strojové učenie, umelá inteligencia, Ontosophie
Anotácia:	Diplomová práca sa zaoberá semi-automatickou konštrukciou ontológií na báze textu. Zatiaľ čo jadrom práce bolo vytvorenie integrovaného systému pre semiautomatické dopĺňanie ontológií o nové inštancie ontologií s inštanciami extraktovanými z textu, práca tiež rozoberá a analyzuje dva hlavné prístupy v tejto oblasti. Systém je založený na báze kontrolovaného učenia a teda ten sa najprv naučí extrakčne pravidlá z anotovaného textu a potom ich aplikuje pre extrakciu z nových dokumentov. Doležitou súčasťou celého cyklu populovania ontológií je užívateľ – expert, ktorý akceptuje, zamieta alebo modifikuje novo extrahované a systémom navrhované inštancie. V práci je taktiež podaná analýza možnosti automatického vytvárania nových tried.

Thesis title:	Semi-Automatic Construction of Ontologies from Text
Department:	Department of Cybernetics and Artificial Intelligence, TU FEI Košice
Author:	Dávid Čeljuska
Supervisor:	Ing. Ján Paralič, Phd.
Consultant:	Dr. Maria Vargas-Vera KMi – Knowledge Media Institute, The Open University, United Kingdom
Date:	9. 5. 2004
Keywords:	system for ontology population, ontology construction, ontology population, instance, text, rule confidence, computation of confidence, natural language, information extraction, machine learning, artificial intelligence, Ontosophie

Annotation: The Master's Thesis deals with semi-automatic construction of ontologies from text. While the core of the thesis was to develop an integrated system for ontology population with instances extracted from text, it also discusses and analyzes two major existing approaches in this area. The system is based on supervised learning and therefore learns extraction rules from annotated text and then applies those rules on newly documents for the extraction. The important part of the entire cycle of ontology population is a user who accepts, rejects or modifies newly extracted and suggested instances to be populated. An analysis of the possibility of automatically creation of new classes is discussed in turn.

Contents

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). They are becoming essential in many on-line applications including Yahoo!, Google, Amazon, and eBay.

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., 2001a; Maedche and Staab, 2000a; 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 construction and maintenance process. Here, the 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.

The thesis is organized as follows. The first part (chapter 1 and 2) gives a definition of an ontology based on AI literature and discusses the various major challenges in the field of ontology construction. An analysis of the two different major trends in this field is given in the following chapter 3. The core of the thesis was to develop a system that is able to perform semi-automatic population of ontologies with instances from text. The developed system, which uses approaches from Natural Language Processing and Information Extraction is described in chapter 6. While the framework of the system was motivated by (Vargas-Vera et al., 2001b) the thesis extends the idea by incorporating rule confidence values to the scene in order to gain higher precision and better performance. Two methodologies for its computation are described in chapter 6.7. In addition to the population of ontology with instances, the thesis also gives an analysis on the possibility of creating semi-automatically new classes from text – chapter 7. The validation of the system and experiments with using rule confidence values either with performing automatic rule elimination or without are given in chapter 8.

1 Definition of ontology and terms

The Artificial-Intelligence literature contains many definitions of an ontology.

The term is borrowed from philosophy, where the ontology is a systematic account of Existence. According to (The Free On-line Dictionary of Computing[1]), an ontology is an explicit formal specification of how to represent the objects, concepts[2] and other entities that are assumed to exist in some area of interest and the relationships that are held among them.

The following definition (Gruber, 1993) is widely used. An ontology is the specification of conceptualizations, used to help programs and humans share knowledge. In this usage, an ontology is a set of concepts/classes - such as things, events, and relations - that are specified in some way, such as specific natural language, in order to create an agreed-upon vocabulary for exchanging information.

A more formal definition, adopted in this thesis, is given by (Maedche and Staab, 2001), an ontology can be described by a 5-tuple consisting of the core elements of an ontology such as concepts/classes, relations, an hierarchy, a function that relates classes non-taxonomically and a set of axioms. Then the ontology $O = \{C, R, H, f, A\}$ consists of:

- C, classes and R, relationships, are two disjoint sets.

[1]FOLDOC - `http://foldoc.doc.ic.ac.uk/foldoc/index.html`

[2]concepts are often called classes and this term is adopted in the thesis

- H, class hierarchy H: $H \subseteq C \times C$ also called *taxonomy*. $H(c_1, c_2)$ means that c_1 is a subconcept or a subclass of class c_2.
- f, a function that relates classes non-taxonomically: $f : R \rightarrow C \times C$.
- A, a set of ontology axioms expressed in appropriate logical language.

Taxonomic relationships are often presented as *is-a* relationships and this notation is used throughout this thesis. Even more, slots are recognized as a set of attributes for a given class. For example a class called *Event* might have slots such as: *has-location, start-time, end-time* and *list-of-participators*. One can notice that the type of slots may vary. While *location* can be some kind of region, *start-time* will certainly be of time or date type. In addition, in the ontology terminology one might come across with instances. An instance is one particular object or instance of some class. For example *SAMI2004*, the conference on Applied Machine Intelligence held in Herľany, Slovakia on 16-17 January 2004, is one particular event and therefore it is an instance of a class *Event*. One might note that slot values for this example are going to be: *location = "Herľany, Slovakia"*, *start-time = 16-01-2004*, *end-time = 17-01-2004*. Figure $1-1$[3] shows part of an *Event* ontology, *SAMI2004* (printed in red) is an instance of class *Conference* which is some kind of *Meeting-Taking-Place*, which is some kind of *Social-Gathering* which is an *Event*. Despite that explicitly from the figure $1-1$ class *Conference* contains only slots *meeting-attendees* and *meeting-organizer,* it also inherits all the slots from its super-classes - that is *start-time, end-time* and *has-location*.

2 Ontology acquisition from text and challenges

As it was said in the previous chapter the understanding of ontologies is different among different Artificial Intelligence (AI) researchers, this therefore causes them to differ in their point of views and approaches.

[3]for the sake of space, only Event and Conference classes were expanded, but every class in the ontology contains predefined slots

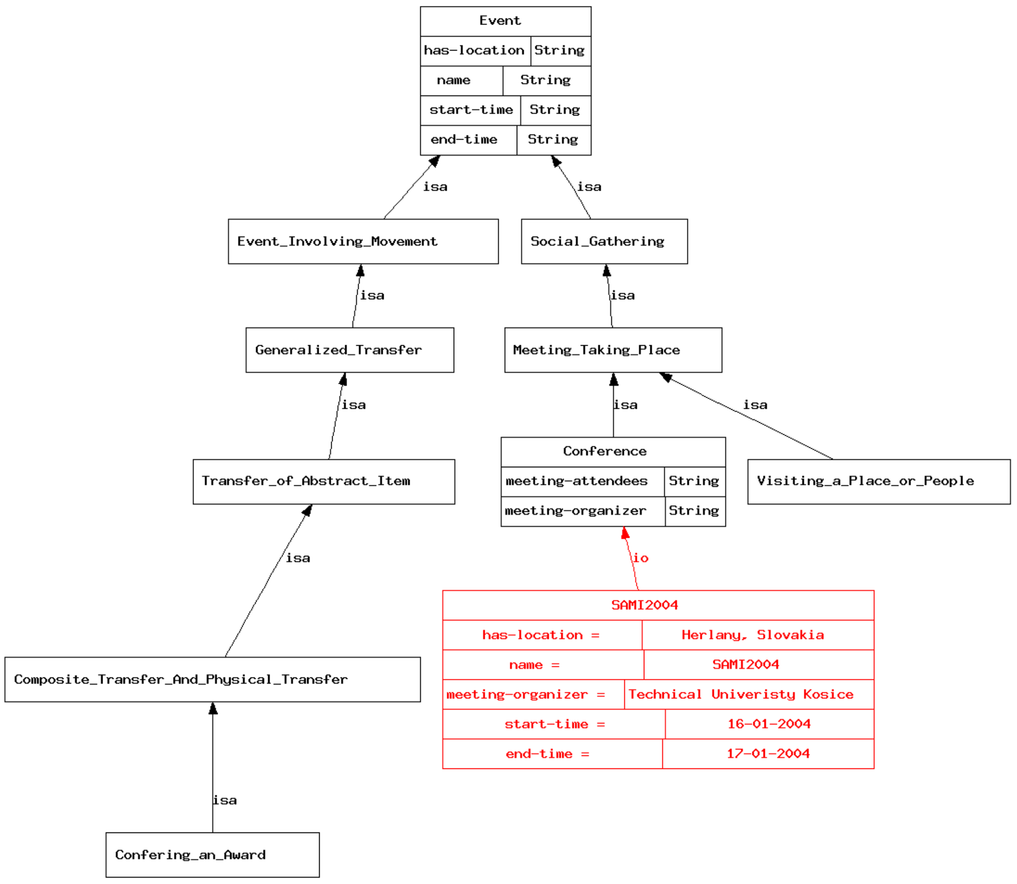

Event
has-location | String
name | String
start-time | String
end-time | String
isa
Event_Involving_Movement
Social_Gathering
Generalized_Transfer
Meeting_Taking_Place
Transfer_of_Abstract_Item
Conference
meeting-attendees | String
meeting-organizer | String
Visiting_a_Place_or_People
io
SAMI2004
has-location = | Herlany, Slovakia
name = | SAMI2004
meeting-organizer = | Technical Univeristy Kosice
start-time = | 16-01-2004
end-time = | 17-01-2004
Composite_Transfer_And_Physical_Transfer
Confering_an_Award

Figure 1 – 1 An example of Event Ontology

However, the basic idea of all of these approaches and views is quite similar and could be simply describe as:

Taking a set of textual documents, running a sophisticated tool and getting an ontology as the result.

While information contained in textual documents could be understood as flat representation, knowledge represented in ontology is hierarchical - what is clear from their definition. The way from flat to hierarchy/from information to knowledge is not so trivial and incorporates many AI fields such as Natural Language Processing, Machine Learning, Information Extraction, Clustering, Classification and so on.

Two ontologies describing the same domain might be completely different and in addition, if two people are given the same set of documents and are asked to outline an ontology, their results will certainly be different. One of them might concentrate on one hierarchical aspect while the other on another one. The question here is what information, concepts and relationships we are mostly interested in from a given set of documents. Therefore, the tool should explicitly know or should be built for a specific task, otherwise it will not be able to determine what is important for us and what is not.

Splitting the basic idea down, we deal with different problems. Some of them are listed here:

- Recognize concepts/classes
- Define slots for each class
- Discover taxonomic relationships
- Discover non-taxonomic relationships
- Extracted ontology versus the truth
- Ontology population with instances
- Ontology refinement and maintenance

The vast number of tasks that need to be undertaken in order to fulfill the basic idea has lead the researchers to concentrate on one aspect at a time. Even more, it became obvious that a fully automated approach, based on the current technology, is not quite feasible[4]. The following chapter analyzes a couple of these approaches in detail.

3 Approaches to ontology construction

This chapter analyzes two different trends in ontology construction. The first one discovers non-taxonomic relations from text and enhances an already taxonomic hierarchy

[4]"Humans do it better" – Open Dictionary

based on association rules. The second one discovers taxonomical relationships and places discovered concepts into hierarchy, based on clustering of sub-categorization frames.

3.1 Text-to-Onto

The first approach (Maedche and Staab, 2000c, 2001) deals with discovering non-taxonomic relationships from text and enhancing already defined taxonomic hierarchy. Their Text-to-Onto (Maedche and Staab, 2000d) system uses shallow parsing as a natural language module. This module consists of tokenizer, morphological and lexical processing and chunk parsing that uses lexical resources to produce mixed syntactic/semantic information. The output of this module is then XML-tagged text.

The learning component is for discovering non-taxonomic relationships[5]. The system is based on discovering generalized association rules (Strikant and Agrawal, 1995). The mining generalized association rules is an extension to mining association rules technique - Apriori algorithm. While the original Apriori algorithm considers all items to be completely disjoint without any hierarchy (*milk* and *bread* are considered to be as similar as *Pepsi* and *Sprite* or *bicycle* and *tea*), the extension towards generalized association rules considers *Pepsi* and *Sprite* to be *soda* and *soda* is considered to be a *beverage*; *milk* and *bread* is *food* while *bicycle* and *tea* share only one class - *item*.

Taking the hierarchy of items might result in getting association not just between instances (items at the lowest level - *"crunchy chips"* and *"diet coke"*) but also between classes (*snack* and *soda*) or between a class and an instance (*"crunchy chips"* and *"soda"*). Therefore, the result of their learning module is a set of couples/classes that is understood as a relationship between them. However, there are two issues that need to be solved. One, is that my small experiment with association rules (Čeljuska, 2003) showed, is that a huge number of irrelevant rules is very high among a small number of interesting ones. Therefore, a good measure is needed in order to classify them. The

[5]formally speaking this would mean function f from $O = (C, R, H, f, A)$

other problem here is, that those relationships are unlabeled. The algorithm extracts some rules between them, i.e. *snack* and *soda* but one does not know what this relationship is nor knows what direction it goes. Having only unlabeled a relationship is not sufficient enough.

In addition to taxonomic relationship that is needed prior to relationship extraction, the system also requires a lexicon. The lexicon defines what class a particular item is part of. For example, instances such as *Pepsi*, *Coke*, *Sprite* are sodas; *Hilton*, *Marriott*, *Balagio* are hotels; *Bancha*, *Earl Gray*, *Chinese powder* are teas. A lexicon is needed in order for the learning module to understand that for example *Marriott* is a hotel.

Although, the entire approach seems to be more effort than gain it has been implemented in a workbench environment with a couple of other approaches to form one ontology construction tool (Maedche and Staab, 2001).

3.2 ASIUM

Asium (Faure and N'edellec, 1998) is able to learn semantic knowledge from text. In this context it means extracting concepts/classes and putting them into taxonomic relationship[6]. It is a semi-automatic system meaning that user's control is needed in the process. Asium learns semantic knowledge and ontologies in the form of sub-categorization frames of verbs. A sub-categorization frame in this context is defined as:

> <verb> <preposition or syntactic role: headword> <preposition or syntactic role: headword> . . .

For example, a sub-categorization frame of the sentence *My father travels by car* is: *<to travel> <subject: father> <by: car>*. The system uses a stop list and it only takes headwords into consideration, So all articles, adjectives, etc. (*a, the, my, your, nice, beautiful, etc.*) are ignored, due to them still being believed to be preserved semantic information. Moreover, syntactic parser Sylex identifies whether headwords

[6]formally speaking the approach concentrates on finding C and H from $O = (C,R,H,f,A)$

are expressions, i.e. *double decker, Ford Escort* or single words. The syntactic parser gives all possible frame interpretations of sentences and ASIUM uses all of them for this approach to avoid a very time consuming hand disambiguation step while still giving a good outcome.

Once each sentence has been instantiated into a frame the learning component takes them as input and learns an ontology. This step incorporates unsupervised clustering (bottom-up) and relies on the following assumption:

> *Headwords occurring after the same preposition or syntactic role, and with the same verbs represent the same concept.*

For example from *<to travel> <subject: father> <by: car>* and *<to travel> <subject: father> <by: train>* one can conclude that *car* and *train* represent the same concept, i.e. *motorized vehicle*.

This assumption is implemented in two steps. The first step gathers headwords that occur in the same contexts such as with the same verb and the same preposition or syntactic role. The second one builds synthetic frames according to verbs of subcategorization frames and assigns number of occurrence in the given context. For example, from the following instantiated frames:

> <to travel> <subject: father> <by: car>
> <to travel> <subject: mother> <by: train>
> <to drive> <subject: friend> <object: car>
> <to drive> <subject: colleague> <object: motorbike>
> <to drive> <subject: friend> <object: motorbike>

Asium might create synthetic frames, one per verb:

> <to travel> <subject: [father(1), mother(1)]> <by: [car (1), train(1)]>
> <to drive> <subject: [friend(2), colleague(1)]> <object: [car(1), motorbike(2)]>

At this stage clustering comes into play. The clustering is based on the distance between two clusters. In this context a cluster is represented as a list of headwords, for example *[car(1), train(1)]*. Overlapping clusters are aggregated into a new cluster. Thus, clusters that contain the same headwords with the same frequencies are considered to be similar - their distance is zero. On the other hand, the distance of clusters that do not share any headword is the highest, equal to 1. The clustering algorithm is very simple and could be briefly described as an examination of each possible couple of clusters, aggregating those pairs that are the most similar[7] and repeating the same step over and over again until it is not able to aggregate any pair anymore. It is important to understand that aggregated are the headwords of two clusters (no frames). For example *[car(1), train(1)]* and *[car(1), motorbike(2)]* might be aggregated to form new cluster called *motorized vehicle*[8]. After the aggregation, the new cluster is propagated through all the synthetic frames, meaning that every occurrence of *[car(1), train(1)]* and *[car(1), motorbike(2)]* will be replaced with *motorized vehicle*. At this stage, a user is asked to accept or reject the aggregation to be propagated. For instance aggregation might yield new cluster *[car(1), train(1), bike(1), motorbike(2)]*. While this cluster is certainly good for a frame *<to travel>*, it is no good for *<to drive>* since everyone knows that a bike is not drivable because it is not a motorized vehicle.

In this description a concept/class of a building ontology is a cluster. Therefore at each level of clustering new classes are introduced. One can observe that clustering, which at each level creates only pairs, might lead to enormous number of useless classes. Asium has however a post-processing phase in which it removes all useless classes.

This approach might be a big help in ontology construction in a narrow specific domain but it might not be very useful in a general one. In case of a frame such as *<to present>*, *<to give>* or *<to perform>* the set of headwords might differ resulting in the user being asked to accept or reject too many aggregations that he will be rejecting.

[7]a threshold value is used for distance pruning

[8]the name itself is given by an user

It has to be said that the method has been adopted in a workbench tool with Text-to-Onto, which is described in the previous chapter (chapter 3.1).

3.3 Summary of existing approaches

In addition to the mentioned systems (Khan and Luo, 2002) describes a system which is based on simple clustering of documents based on modified self-organizing maps (SOM) with the extension of topic tracking. The system is then capable of clustering documents and labeling clusters. It also uses Wordnet[9], a general ontology lexicon database, as a tool for labeling.

The system can be useful in the first stage of ontology construction because it can generate the first seed of an ontology.

At this point one can understand that the topic of ontology construction is vast. Thus one system might concentrate on one aspect, another might on a different one. Therefore, it is not easy to compare them as they do not share a lot in common. Analyzed approaches only cover a part of ontology construction process. While Text-to-Onto (Maedche and Staab, 2000d) concentrates on discovering non-taxonomic relationships, Asium (Faure and N'edellec, 1998) is designed to build taxonomic relationships.

4 Ontology population and information extraction

An ontology population with instances is one of the issues the ontology construction, acquisition and maintenance addresses. In this task, it is assumed that an ontology with both taxonomic and non-taxonomic relationships has more or less been constructed. In this scenario, the goal is to feed classes with relevant instances and perform necessary rearrangements of classes if required.

The ontology population from text, more over, assumes that a given ontology will be populated with instances extracted from natural language text such as plain text or HTML

[9]Wordnet - `http://www.cogsci.princeton.edu/∼wm`

for example.

In the thesis each ontology class C_i is defined as a tuple $C_i = (n_i, L_i)$, where n_i is a lexical name of the *i*-th class and L_i is a vector of couples (s,t), where s is a lexical name of slot and t is its type. Therefore $C_i = (n_i, ((s_1,t_1),(s_2,t_2),\ldots,(s_{Ni},t_{Ni})))$ and $Ni = card(L_i)$ is size of L_i (number of slots in class C_i). For example, class *Conference* from *Event Ontology* example (figure $1-1$) will be formulated as: C_i = *(conference, ((has_location, String), (start_time, Date), (end_time, Date), (meeting_attendees, String), (meeting_organizer, String)))*. One can notice that some slots are inherited from all the ancestors from the taxonomic hierarchy.

Instance I_{ij} is then defined as a tuple $I_{ij} = (m_i, V_i)$, where m_i is a lexical name of the instance and $V_i = (v_1, v_2, \ldots, v_{Ni})$ is vector of its slots' values. For example, the instance *SAMI2004* from the example figure $1-1$ is: I_i = *(SAMI2004, (Herľany, Slovakia, 17-01-2004, 18-01-2004, N/A, Technical University Košice)*.

Given a set of textual documents of a particular domain described by the given ontology, the task of ontology population is to extract a set of instances for classes $C_1, C_2, \ldots, C_i, \ldots, C_M$ and feed them into the ontology. This, in particular means, to extract all the entities (slot values - $v_1, v_2, \ldots, v_{Ni}$) from the text for a given class C_i. At this point Information Extraction comes into play.

4.1 Information extraction as a tool to ontology population

Information Extraction (IE) is the task of obtaining structured information from unstructured sources i.e. natural language text (Engels and Lech, 2003). Generally speaking, it is the extraction of pertinent information from a large volume of data or text. Information Extraction recognizes two different strategic approaches. On the one hand, there are systems using goal-driven (top-down) approach and, on the other hand, there are also initiatives extracting information in content-driven (bottom-up) manner (Engels and Lech, 2003).

Goal-driven systems are more domain specific than content-driven. Information of interest is typically predefined in a form of empty templates/slots/extraction rules. For example, a template *X visited Y* might be used to extract two entities, such as visitor *X* and, for example, a place *Y*. Those systems are pretty precise (MUC - Series of Message Understanding Conferences).

On the other hand, content-driven systems are designed to work more domain-independently. Although, this makes them much less precise, in terms of the number of correctly extracted entities, they become important when it comes to text summarization and agent-based searching (Engels and Lech, 2003).

One can now see that goal-driven IE systems might become an essential part of Ontology Population Systems. Those Information Systems are able to extract important and relevant entities from text. Then those entities become specific slot values - vector V_i.

In order to get high precision[10] a good quality set of extraction rules is needed. However, automatic generation of extraction rules is not an easy task as not only is it domain specific but also depending on the writers style. This implies that a good Text-Preprocessing and Linguistic Analysis has to be done beforehand.

5 Linguistic analysis

Linguistic Analysis incorporates number of steps and the field itself is very complex. Most of the systems dealing with natural language text recognize some of the following steps:

- Tokenization
- Lexical Analysis
- Morphological Analysis
- Syntactical Analysis

[10]Precision is defined as $P = c/n$, where c is number of correctly extracted entities and n is total number of extracted entities

Some systems might extend this list and others might perform particular steps not in the given order.

In a nutshell, tokenization deals with recognizing sentences and its boundaries. Although, this task might seem very trivial as its looking for uppercase starting letters and then period, it is not that simple. Some systems even use statistical models to deal with the full stop versus abbreviation period problem. More over, in some cases sentences are split down into chunks (Marmot)[11].

Lexical Analysis in most cases means tagging tokenized text with part-of-speech (POS) tags. This step is essential in the case of determining between grammatical entities such as articles (*the*, *a*), pronouns (*she*, *he*, *we*) and so on, and lexical words such as verbs (*give*, *won*), nouns (*award*, *conference*) and adjectives (*serious*, *highly prestigious*).

Morphological and Syntactical Analysis are important parts in the process as they help to determine right morphological category as much as syntactical. For example, to determine whether *"post"* is a noun (*"a post"*) or a verb (*"to post"*). The outcome of this stage is in most cases tagged text with lexical and grammatical categories.

6 System for semi-automatic population of ontologies with instances

Designed system, Ontosophie, in which its framework was motivated by Info-Extractor (Vargas-Vera et al., 2001a,b) 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 and HTML. Therefore, as it was stated in the previous chapter, the task is to identify as many possible entities $v_1, v_2, \ldots, v_{Ni}$ and thus to construct a vector V_{ij} 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 instances I_{ij} for class C_i are correct and whether they should be fed into it. This

[11] Marmot is described in chapter 6.5

determination is based on the extracted entities and their confidence, where computation of them is described in chapter 6.7.1.

For better explanation of the system's architecture it will be described by using a particular example given in the following chapter together with assumptions that are necessary for explanation.

6.1 Description of the ontology

Experiments were performed by using KMi's[12] Event ontology O. This consists of events or activities that are defined formally in the ontology as classes $C_i : i \in \{1, 2, \ldots, M\}$. Currently, the KMi ontology contains 41 different types of events/classes ($M = 41$). A small part of the ontology is shown in figure $6-2$[13] to aid necessary understanding of an ontology. Each class/event C_i is defined with set of slots $s_1, s_2, \ldots, s_{Ni}$, which might be instantiated by an information extraction component. Type of each slot is in default *String* ($t_1 = t_2 = \ldots = t_{Ni} = String$), 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 three different classes from the event topology in order to explain their structure:

```
Class Event 1:  Visiting-a-Place-or-People
Description:  Class of an event where someone visits some place or someone
else.
Slots:
visitor (list of persons)
people-or-organisation-being-visited (list of persons or organizations)
has-duration (duration the visit took)
start-time
end-time
```

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

[13] Classes that this thesis refers to are highlighted

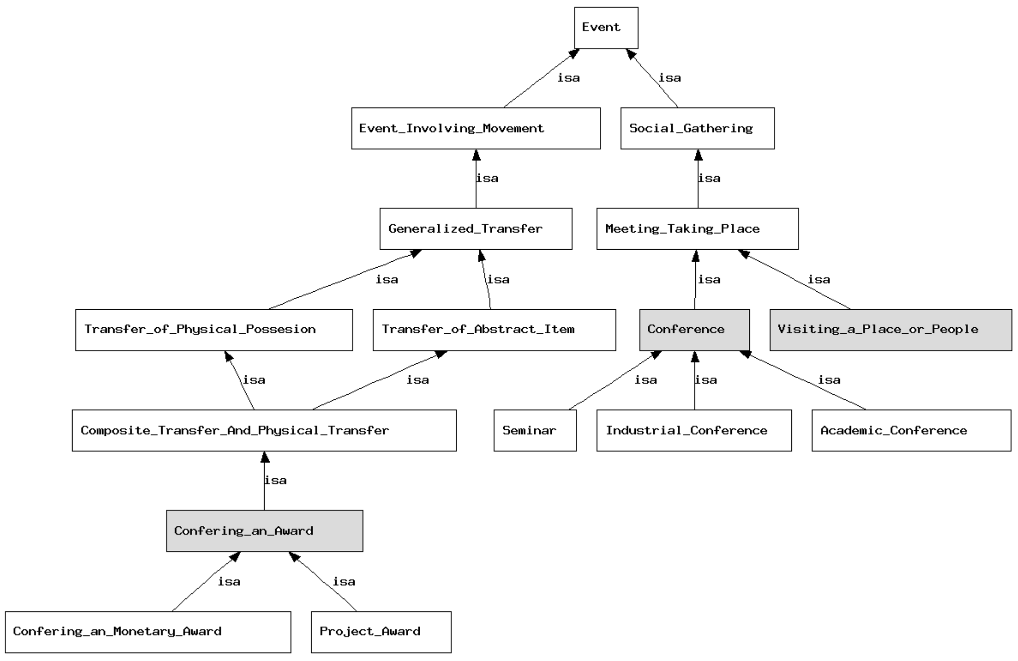


Figure 6 – 2 A Part of Event Ontology

```
has-location (a place where it took place)
```

The structure of Event 1 *Visiting-a-Place-or-People* describes a set of entities, that might be encountered in an article describing an event talking about a visit such as *visitor*, *people-or-organisation-being-visited*, *location*, and so on.

```
Class Event 2:  Conferring-an-Award
Description:  Class of an event describing an event of awarding someone
Slots:
has-duration (when or how long the event took)
start-time (when the event started)
end-time tag:  (when the event was over)
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)
```

The structure of Event 2 describes a set of entities that might be encountered in an article describing an event of awarding some organization or individual. For example donor, recipient agent, amount of money and so on.

```
Class Event 3:  Conference
Description:  Class of an event that describes a conference, workshop
or seminar event.
Slots:
has-duration (duration)
start-time
end-time
has-location (a place it took place at)
main-agent (name of the conference, workshop, seminar)
meeting-attendees (list of persons who took part of the conference)
meeting-organizer (organization that organized the event)
```

In the case of Event 3 entities such as place, name of the seminar are important entities that the system is looking for in the document that is being instantiated.

6.2 System's framework

Based on the assumptions and the given example stated in the previous chapter the description of the system's framework consists of the following steps (figure 6 – 3):

- Annotation/Mark-up
- Learning

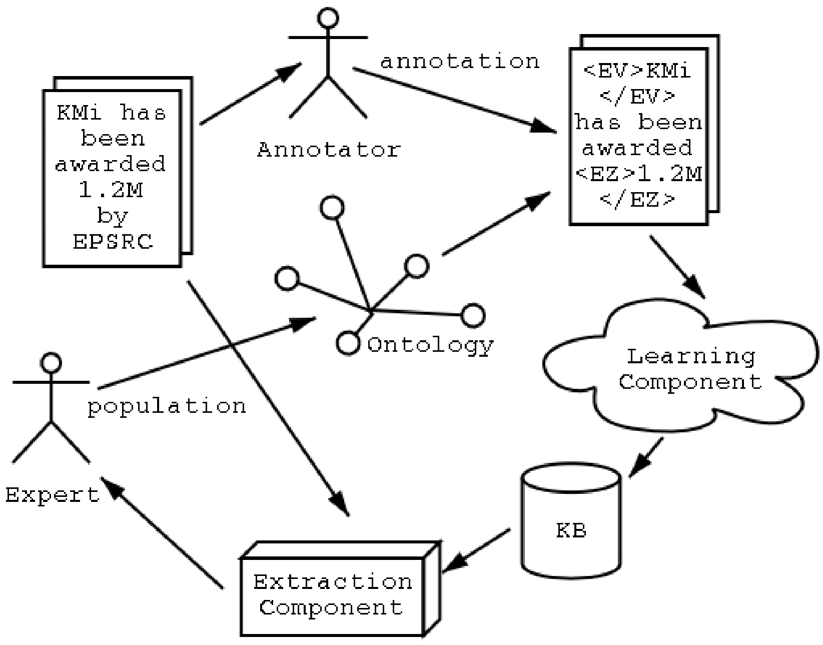


Figure 6 – 3 The framework of the system

- Extraction
- Population

Each of these steps will be explained in turn.

6.3 Annotation

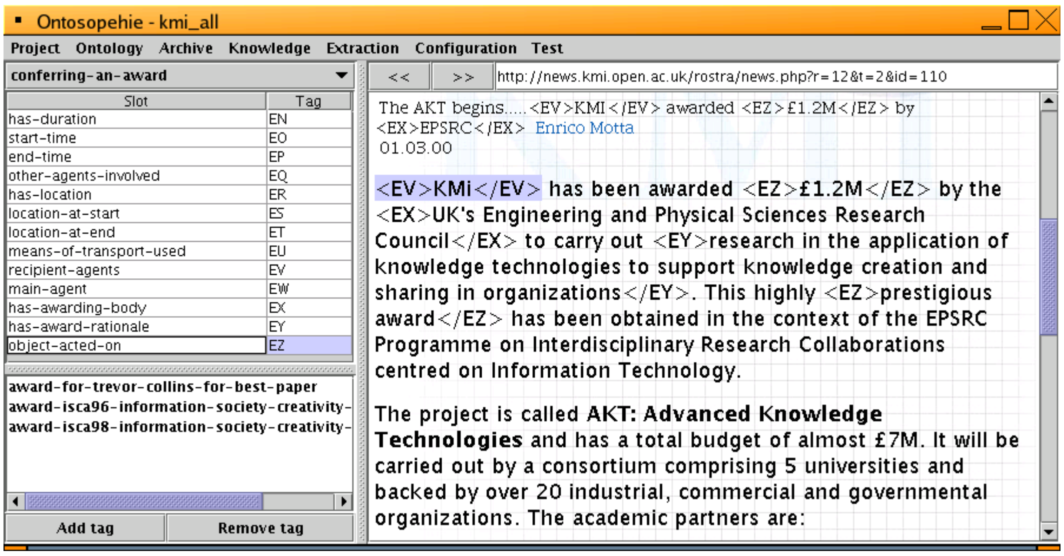
The activity of semantic tagging refers to the activity of annotating text articles (written in natural language in plain text or HTML) with a set of tags defined in the ontology.

Each slot that occur within any class of the ontology is assigned a unique XML tag. If no tag is assigned to any slot within an ontology the system gives an option to generate unique XML tags for each slot automatically. In this case the ontology with assigned tags can be saved for later use.

Ontosophie has a basic Internet browsing facility (figure 6 – 4), which contains navigation buttons ("*< Back*" and "*> Forward*") and an entry field for entering a url.

The annotation component extracts sets of possible tags for a given class defined in the ontology. Therefore, the 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 – (left-hand side ontology panel figure 6 – 4). During this phase the user inserts relevant XML tags into the document. The system has also an option of removing tags as well. An annotated article might then look as follows:

> The AKT begins.....**<EV>KMI</EV>** awarded **<EZ>L1.2M</EZ>** by **<EX>EPSRC</EX>**
>
> Enrico Motta 01.03.00
>
> **<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>**. This highly **<EZ>prestigious award</EZ>** has been obtained in the context of the EPSRC Program on Interdisciplinary Research Collaborations centered on Information Technology. . .


Ontosopehie - kmi_all
Project Ontology Archive Knowledge Extraction Configuration Test
conferring-an-award
<<
>>
Add tag
Remove tag

http://news.kmi.open.ac.uk/rostra/news.php?r=12&t=2&id=110

Slot	Tag
has-duration	EN
start-time	EO
end-time	EP
other-agents-involved	EQ
has-location	ER
location-at-start	ES
location-at-end	ET
means-of-transport-used	EU
recipient-agents	EV
main-agent	EW
has-awarding-body	EX
has-award-rationale	EY
object-acted-on	EZ

award-for-trevor-collins-for-best-paper
award-isca96-information-society-creativity-
award-isca98-information-society-creativity-

The AKT begins.....<EV>KMI</EV> awarded <EZ>£1.2M</EZ> by <EX>EPSRC</EX> Enrico Motta
01.03.00

<EV>KMi</EV> has been awarded <EZ>£1.2M</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>. This highly <EZ>prestigious award</EZ> has been obtained in the context of the EPSRC Programme on Interdisciplinary Research Collaborations centred on Information Technology.

The project is called **AKT: Advanced Knowledge Technologies** and has a total budget of almost £7M. It will be carried out by a consortium comprising 5 universities and backed by over 20 industrial, commercial and governmental organizations. The academic partners are:

Figure 6 – 4 Annotation phase

Looking at figure 6 – 4 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.

Each time an article is annotated it has to be saved locally for later use. It is important to know that before the document is saved a simple text-preprocessing is performed. The purpose of this is, that Ontosophie works with HTML files that might contain some characters in Unicode (UNC-8 which is 16 bites long), i.e. pound sign £. While, in most cases, OS does not support Unicode, it has to be converted to a 8-bit long code. In particular, the system converts it into ISO-8859-1 for Western Europe as the system was designed for the English language. For example a pound sign £ is converted into uppercase *L*. The detailed information on converting is provided in the User Guide.

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

6.4 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 (chapter 6.3).

Learning consists of two steps:

- Natural language processing
- Learning extraction rules

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

6.5 Natural language processing

Natural language analysis is extremely crucial step 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[14] natural language processing system.

Marmot (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. Marmot includes the following steps:

- Preprocessing abbreviations to guide sentence segmentation (Mr., Mrs., Dr.) and to solve problem of full stop versus abbreviation period (chapter 5)
- Resolving sentence boundaries based on previous step
- Identification and reformatting parenthetical expressions - grammatically independent entities such as brackets and enumerations
- Recognition entities from a phrasal lexicon (in order to, as well as, inasmuch as)
- Recognition of date and duration phrases
- Performing part-of-speech tagging (POS)
- Identification of nouns, prepositions and adverbial phrases
- Scoping conjunctions and disjunctions (and, or)

However, before Natural Language Processing begins the system performs text-preprocessing. At this stage, Ontosophie offers a user to take control over this by running a user defined script for a given set of annotated files[15]. This gives high robustness to the system. The script can be written in any language (Perl, Sed, Awk, Java) as long as the Users Operating System is able to interpret it. The most important parts of the script should at least include the following:

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

[15]User Guide gives more detailed information on this

- Normalizing dates into MM-DD-YYYY format as Marmot expect dates to be in the U.S. and not in European style (DD.MM.YYYY).
- Additional sentence boundary normalization. This should include putting full stop periods at the end of each enumeration sentence, titles and so on. This step is very important.

At this stage Marmot is ran. Marmot's output for annotated example given in chapter 6.3 will look as follows:

```
<ex> 1 1
SUBJ(1): THE AKT
VB  (2): BEGINS
PUNC(3): %PERIOD%
</ex>

<ex> 2 1
SUBJ(1): KMI          <EV>
VB  (2): AWARDED
OBJ1(3): 1.2M          <EZ>
PP  (4): BY EPSRC          <EX>
PUNC(5): %PERIOD%
</ex>

<ex> 3 1
SUBJ(1): ENRICO MOTTA
ADVP(2): @03-01-00@
PUNC(3): %PERIOD%
</ex>

<ex> 4 1
SUBJ(1): KMI          <EV>
VB-P(2): HAS BEEN AWARDED
OBJ1(3): L1.2M          <EZ>
PP  (4): BY THE UK %POSS% ENGINEERING AND PHYSICAL SCIENCES RESEARCH
```

```
COUNCIL
<EX>
</ex>

<bad_ex> 4 2
VB  (1): TO CARRY OUT
OBJ1(2): RESEARCH          <EY>
PP  (3): IN THE APPLICATION OF KNOWLEDGE TECHNOLOGIES           <EY>
</bad_ex>

<ex> 4 3
VB  (1): TO SUPPORT
OBJ1(2): KNOWLEDGE CREATION
</ex>

<ex> 4 4
ADVP(1): AND
VB  (2): SHARING
PP  (3): IN ORGANIZATIONS
PUNC(4): %PERIOD%
</ex>
```

One might notice, that each sentence was split into segments (*<ex>*. . . *</ex>*). The part between *<bad_ex>*. . . *</bad_ex>* tells that the entire text that was tagged with *<EY>* (chapter 6.3) tag as one whole piece ("<EY>*RESEARCH IN THE APPLICATION. . . TO SUPPORT KNOWLEDGE CREATION*</EY>") of information was divided into more segments ("4 2" and "4 3") and thus consistency was broken. In the output specific entities within a sentence are identified such as *SUBJ* - subject, *PP* - prepositional phrase, *OBJ* - object, *ADVP* - adverb, *PUNC* - punctuation symbol and so on. Because dates were normalized prior to the analysis, Marmot was also able to recognize them as such.

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

6.6 Generating extraction rules

Learning extraction rules from an annotated set of documents is a task of generating a set[16] of extraction rules. Having the right extraction rules that extract correct and important entities[17] from annotated and pre-processed text is crucial as the precision of extraction phase relies on them. This task refers to Information Extraction technique (chapter 4).

Ontosophie in this phase uses Crystal, a dictionary induction system. A small introduction to Crystal is given in the following part to allow the necessary understanding of this system.

6.6.1 Introduction to Crystal

Crystal[18] (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. The following part shows one of a possible concept node generated by Crystal:

```
CN-type project-award   ID:      516
        Status: GENERALIZED
Constraints:
  VB-P::
        mode: passive
        root: awarded
        terms: BEEN AWARDED
        mod terms: BEEN
        head terms: AWARDED
        classes:                ws_Root_Class
        mod class:              ws_Root_Class
        head class:             ws_Root_Class
  OBJ1::
```

[16]also called dictionary

[17]those that are defined as slots within some class in a given ontology

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

```
        mod terms: <null>
        classes:                ws_Root_Class
        mod class:              ws_Root_Class
        head class:             ws_Root_Class
  PP::          ==> has-awarding-body
        terms: BY
        mod terms: BY
        classes:                ws_Root_Class
        mod class:              ws_Root_Class
        head class:             ws_Root_Class
Coverage:    5 Errors:    1
```

One might notice, that the rules purpose is to extract an entity of type *conferring-an-award*, which in this case refers to name of a class from the ontology O (figure$6-2$). This concept node (CN), 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 as long as it does not contain a word modifier such as *"the"*, *"a"*, *"prestigious"* and it contains 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 ..."*[19] 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[20].

[19]this is one of the sentence from the annotated article described in chapter 6.3

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

Crystal, firstly, initializes a rule dictionary with CN definitions for each positive training instance. In this context a training instance is one piece of the word construct either annotated or not. These initial definitions are designed to extract the relevant phrases from the training instance from which they were initiated, but are overly specific to apply to previously unseen data. The main task of Crystal is to gradually relax the constraints in the dictionary and to broaden their coverage, while merging similar definitions to form a more compact dictionary (Soderland at al., 1995).

The generalizations of its initial concept node definitions is continuing as long as it covers all the positive instances while not covering any negatives[21]. The similarity between rules is deducted by counting the number of relaxations required to unify two concept node definitions. In case any unifications takes place it is tested back on the training data to make sure it does not cover any negative example (informally speaking, non-marked-up entity). If the new unified concept node is valid, Crystal removes all the instances covered by the new CN and inserts newly created CN to the dictionary. The routine is repeated. If eventually, a point is reached where further relaxation would lead to a CN that exceeds some pre-specified error tolerance Crystal begins the same process with a different CN until all initial CN have been considered for generalization (Soderland at al., 1995).

Crystal unifies two similar definitions by finding the most restrictive constraints that covers both. If word constraints from the two CN have an intersecting string of words, the unified word constraint is that intersection's string.

Unifying two class constraints may involve moving up the semantic hierarchy to find a common ancestor for classes in the two constraints. Class constraints are removed entirely when they reach the root of the semantic hierarchy.

In order for Crystal to know semantic hierarchy, mentioned above, it has to be provided as a priori information in two files. One of the file contains semantic class hierarchy definition such as:

```
ws_Root_Class                    ws_Root_Class
```

[21]this is the basic idea of specific-to-general algorithm from Machine Learning

```
# kmi classes
ws_KMI                          ws_Root_Class
ws_Institute_Organization       ws_KMI
ws_University                   ws_Institute_Organization
ws_Department                   ws_Institute_Organization
ws_Institute                    ws_Institute_Organization
ws_Council                      ws_Institute_Organization
```

From which Crystal knows that, i.e. *ws_University* is more specific concept of *ws_Institute_Organization*.

The other file contains lexicon where for each entity is defined to what class it is assigned to. Just for a short look-in a part of the lexicon that was used for experimentation (chapter 8) is presented:

```
OULU_UNIVERSITY                 ws_University
UNIVERSITY_OF_TORONTO           ws_University
MEDICAL_RESEARCH_COUNCIL        ws_Council
TECHNICAL_UNIVERSITY_OF_KOSICE  ws_University
```

In the concept node example above (beginning of this chapter), one might notice *"ws_Root_Class"* next to each term. This in particular means that the term might be part of any *ws* concept or none. However, in some cases Crystal might generate rules/concept nodes with more restricted terms, i.e. *ws_University* which would imply that the term has to be some kind of university.
Getting extraction rules by using Crystal is not sufficient as one rule might have higher confidence than another. Thus, computing rule confidence becomes essential.

6.7 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[22]. 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[23]. The rule confidence tells how sure the system is about the correctness 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, 1996b) showed that precision moves to high 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.

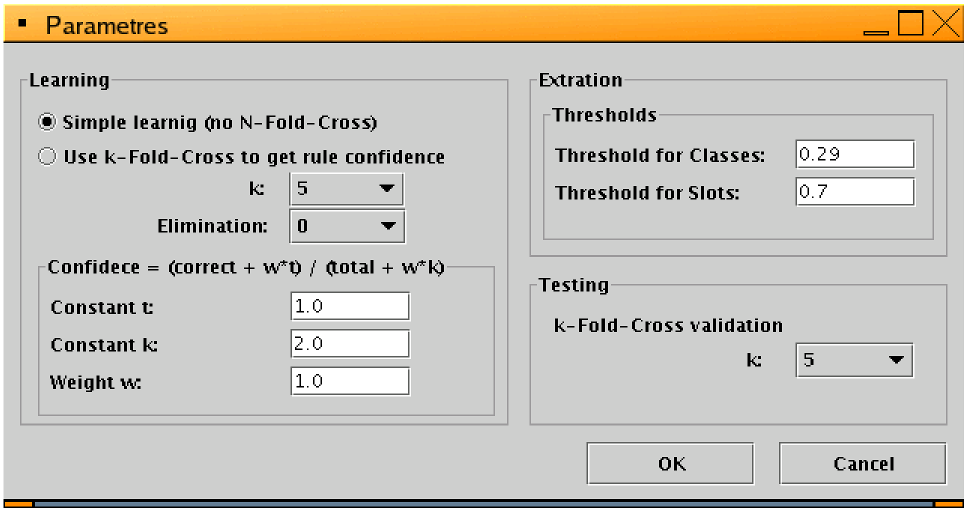
6.7.1 Two ways to confidence of extraction rules

The system is able to use two different ways of getting rule confidence values, which will be assigned to each extraction rule in the dictionary. Figure 6 – 5 shows a dialog window where the preferred method can be chosen (*"Simple learning"* versus *"k-Fold-Cross validation"*).

The first and most simple method uses *Coverage* and *Error* values that are automatically provided for each rule by Crystal (chapter 6.6.1). In this case the rule confidence is computed as:

[22] 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

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


Parametres

Learning

[✓ Simple learnig (no N-Fold-Cross)]

Use k-Fold-Cross to get rule confidence

k:	5
Elimination:	0

Confidece = (correct + w*t) / (total + w*k)

Constant t:	1.0
Constant k:	2.0
Weight w:	1.0

Extration

Thresholds

Threshold for Classes:	0.29
Threshold for Slots:	0.7

Testing

k-Fold-Cross validation

k:	5

Figure 6 – 5 Setting learning and extraction parameters

$$C = \frac{c}{n} = \frac{Coverage - Error}{Coverage} \tag{6.1}$$

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, (6.1) 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 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:

$$LaplaceAccuracy = \frac{n_c + 1}{n_{tot} + k} \tag{6.2}$$

where:[24]

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

- 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 6.2 to the valuation of confidence is then:

$$C = \frac{c+1}{n+2} \tag{6.3}$$

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 $1/2 = 0.5$ [25]. 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.

Further more, (6.2) and (6.3) are generalized into (figure $6-5$):

$$C = \frac{c+wt}{n+wk} \tag{6.4}$$

This gives a user ability to take control over the Laplace equation. If weight $w = 0$ then (6.4) turns into (6.1). The parameter t should in most cases be set to 1 and by itself does not have any deeper meaning – it only gives a user control over the equation along with w (figure $6-5$).

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. It is the methodology for estimating the accuracy of an inducer by dividing the data into k mutually exclusive subsets/folds of approximately equal size. The inducer is then trained and tested k times.

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

Each time it is trained on the data set minus a fold and tested on that fold. The accuracy estimate is the average accuracy for the k folds.

For better explanation lets assume that the system deals with three different classes such as *visiting-a-place-or-people* (A), *conferring-an-award* (B) and *conference* (C) - figure 6 – 6 and three-fold-cross validation is performed. Certainly the sizes of training sets (number of documents) are different for each class. Each class is therefore split into equally sized subsets i.e. $A = A1 \cup A2 \cup A3$. This means, that at each run (1, 2, 3) two of the subsets are taken and the third one is kept for validation. In this particular example, Crystal (chapter 6.6.1) is firstly run for the set $A1 \cup A2 \cup B1 \cup B2 \cup C1 \cup C2$ and validated with $A3 \cup B3 \cup C3$. Then the learning is run for $A1 \cup A3 \cup B1 \cup B3 \cup C1 \cup C3$ and validated with $A2 \cup B2 \cup C2$ and similarly for the third run. One might argue that splitting sets A, B and C by number of documents is not accurate since the methodology says to split sets into approximately equally sized subsets. Certainly, one document might contain more or less positive example (annotated entities) than another. However, statistically speaking, it is not very relevant and splitting them by documents is much more simple.

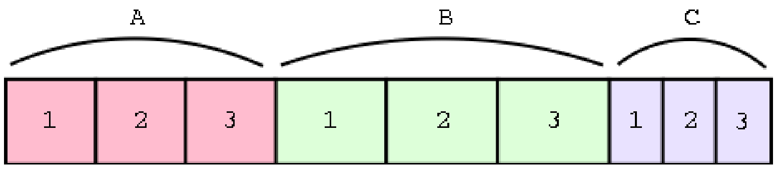


Figure 6 – 6 Three Fold-Cross validation

At each run a new set/dictionary of extraction rules is generated by Crystal. The algorithm 1 outlines the methodology that Ontosophie uses to overcome this problem. The algorithm computes for each rule r_i how many times it is fired correctly c_{r_i}, how many times it fires 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 $n_{r_{new}} = n_{r_i} + n_{r_j}$. Even more a value $x_{r_{new}}$, which tells how many

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**
    $R \leftarrow generateExtractionRules(T)$ {generates a set of rules by running Crystal for set $T$}
    $R \leftarrow setXtoZero(R)$ {sets $x$, number of time the rule was merged, to zero for each rule in the set $R$}
    $R_e \leftarrow evaluate(R,V)$ {$R_e$ is set of evaluated $R$ rules with $V$}
    $W \leftarrow W \cup R_e$
**end for**
**while** $\exists\, i, j, i \neq j; r_i, r_j \in W : constrains(r_i) = constrains(r_j)$ **do**
    $r_{new} \leftarrow merge(r_i, r_j)$ {$constrains(r_{new}) \leftarrow constrains(r_i) = constrains(r_j)$}
    $c_{r_{new}} \leftarrow c_{r_i} + c_{r_j}$ {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**
```

times the rule r_{new} was merged is computed. After the whole process is then:

$$\begin{aligned} c_{r_{new}} &= \sum_{\forall i,i;i \neq j: constrains(r_i)=constrain(r_j)} c_i + c_j \\ n_{r_{new}} &= \sum_{\forall i,i;i \neq j: constrains(r_i)=constrain(r_j)} n_i + n_j \end{aligned}$$

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. The extraction is performed with documents from the validation set on a one by one base. A document from the validation set is used for the extraction (chapter 6.8). 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 and the chapter 6.9 gives detailed information on that.

The rule confidence for each rule at each run is then similarly to (6.4) computed as:

$$C_i = \frac{c_i + wt}{n_i + wk} \tag{6.5}$$

6.8 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[26] and feed a newly created instance into given Ontology O. The document is pre-processed with a user defined script and then with Marmot (similarly as described in chapter 6.5) prior to extraction itself.

[26]not yet processed nor annotated document

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[27] 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 6.8 shows a part of original text[28] 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(figure $6-5$).

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

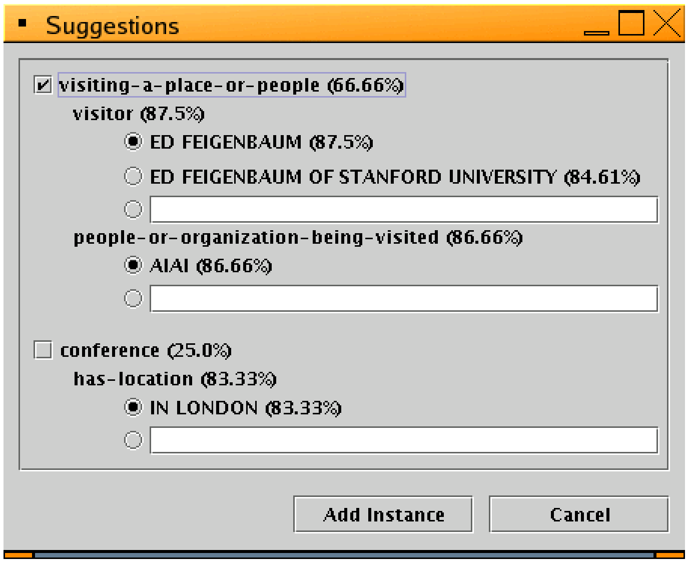
For the information extraction a third component called Badger[29] (Badger and Crystal) was also integrated into the system. Badger makes the instantiation of templates. The main task of Badger is to take each sentence from the document and see if any of the learnt rules can be applied (chapter 6.6.1). If no extraction rule applies to a sentence,

[27] an instance consists of slots and its values (extracted entities)
[28] automatically recognized entities were printed in **bold**.
[29] 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%)
○

Figure 6 – 7 A part of original text and a dialog with extracted entities

then no information is being extracted - irrelevant text is processed very quickly. The following part gives a fast look-in into Badger's output:

```
<cn>    ID: 769 761 Type:           visiting-a-place-or-people
        docid           =          (null)
        sentence_num    =          1
        segment_num     =          1
        visitor ==> SUBJ: ED FEIGENBAUM
</cn>
<cn>    ID: 761 758 Type:           visiting-a-place-or-people
        docid           =           (null)
        sentence_num    =          7
        segment_num     =          1
        visitor ==> SUBJ: ED FEIGENBAUM OF STANFORD UNIVERSITY
</cn>
<cn>    ID: 32         Type:           conference
        docid           =          (null)
        sentence_num    =          10
        segment_num     =          1
        has-location ==> PP: IN LONDON
</cn>
```

This part of the output tells that Badger extracted three entities *“ED FEIGENBAUM”*, *“ED FEIGENBAUM OF STANFORD UNIVERSITY”* and *“PP: IN LONDON”* from yet an unclassified document. Even more, first two entities were recognized as *visitor*[30] for the class *visiting-a-place-or-people* and the last entity was recognized as *has-location* for the class *conference*. In addition, the first value (*“SUBJ: ED FEIGENBAUM”*) was extracted by rule 769 and 761, the second by 761 and 758 and third only with 32. This is an important key to identify back what constraints the rules have, and most importantly what confidence C is attached to it. Once Ontosophie identifies each rule that fires, it 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 chapter will describe solutions given by Ontosophie.

6.8.1 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 (6.4) or (6.5).

[30]*visitor* is specific slot of class *visiting-a-place-or-people*

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.

$$C_{value} = \max_{\forall i: r_i \text{ is in collision}} C_i \tag{6.6}$$

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, $C_{slot} = \max_{\forall i: C_{value,i} is in collision} C_{value,i}$. The highest scored value/entity for a given slot is then pre-selected and values/entities are ordered by their confidence (figure 6.8).

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 6.8). It is important to determine which classes the new instances should be fed into. For this purpose, the class confidence value is assigned to each class. However, it is not easy to perform this decision. Three ways of this computation are presented as follows.

Lets assume that n_{class} is a number of different slots the system is able to fill in with extracted entities for a class $class$. One of the simplest methods is to prefer those classes that the system extracted the most slots for - with the highest n_{class}.

The other approach taking normalization into care, could be in preferring those classes with the highest C_{class} computed as:

$$C_{class} = \frac{n_{class}}{m_{class}} \tag{6.7}$$

Where m_{class} is the total number of slots the system for a given class $class$ could theoretically extract. In other words m_{class} is a number of unique tags throughout all annotated documents of one class.

Third approach is a generalization of the previous one. Instead of taking n_{class} it sums all the slots confidence values[31]. Thus, it prefers those classes that have the highest C_{class}

[31]only slots that fit slot threshold value are taken

computed as:

$$C_{class} = \frac{\sum_{\forall\ extracted\ slots\ for\ the\ given\ class} C_{slot,i}}{m_{class}} \tag{6.8}$$

Experiments have been performed to check which one of those three methods works the best. Details are given in chapter 8. However, it seems that the last two approaches work fairly well.

6.8.2 Pruning

In the previous chapter it was said that the system uses threshold values to distinguish between possibly good and possibly not correct.

There are two different threshold values (figure $6-5$). 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 (chapter 6.8.1). 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 might be 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 re-select 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.

6.9 Validation

The extracted validation entities are important for two different tasks:

- Computation of rule confidence – only in case of k-Fold-Cross methodology
- Validation of the entire system

The process of checking whether the extracted entity (slot) is correct or not is the same for both of the tasks. However, the aim is different.

The computation of rule confidence was described in chapter 6.7.1 and process of validation of the entire system will be given next.

Suppose the system extracted *recipient-agent = "BY THE UK%POSS% ENGINEERING AND PHYSICAL SCIENCES RESEARCH COUNCIL"* for class *conferring-an-award*. In order to determine whether this extraction is correct or not the system has to be provided with an annotated version of the document that was used for the extraction. From the ontology *O*, as it is described in chapter 6.3, the system is able to pull out an XML tag that is assigned to this slot (for *recipient-agent* it is *<ev>*). This is the key to determine whether the extraction was correct or not. However, the task is not as easy as it seems. Any trials to use the exact matching (compare the tagged text and extracted entity) failed. The outcome from the extraction might be different than the tagged entity in the annotated document. One might note, that Marmot[32] converts all the letters into UPPERCASE. Even more it attaches its own tags there (*%POSS%*). In addition, it might happen that the annotator tagged parts separated by commas individually while the extraction got all parts in one. While the extracted information should still be considered to be correct, in case of exact matching it will not be. The list of possible differences between extracted and tagged entities is vast. Ontosophie however is equipped with the solution.

Instead of comparing extracted and tagged pieces in plain text, it compares them after natural language processing was run. Therefore, in order to check an extracted entity both annotated and plain text versions of the document are pre-processed as described in chapter 6.5. As it was mentioned in chapter 6.6.1, Crystal in its extracted output gives the information from what sentence (*sentence_num*) and segment (*segment_num*) the particular entity was extracted from. Once the annotated version of the document is

[32]a natural processing tool (chapter 6.5)

pre-processed the system looks into Marmot’s output and looks up the particular sentence-segment. At this point, the system looks up exactly the same piece of text as the extracted one. In case it has an appropriate XML tag attached to it, then the system knows that the extracted entity is correct.

This technique is one hundred percent precise since the system only performs exact matching but it also has some disadvantages.

When this is applied in order to calculate rule confidence value (chapter 6.7.1) then the system is forcing to extract only those pieces of text that were annotated and from the context they are in. The problem here is, that the system might extract a truly good entity but from context in which it was not annotated. In this case the extraction rule is penalized incorrectly. In addition, when the approach is applied to the whole system validation and the set of documents was not annotated in enough care (all of the entities are identified and annotated) then the gotten precision might falsely go low for the same reason.

7 Analysis of possibility to create new classes

At the current stage, Ontosophie only deals with population of existing ontology with instances gathered from text. However, in some cases the existing ontology might not be sufficient to cover all new coming documents for the extraction in details. In this case it might be valuable to have a system that would be capable of automatically giving suggestions to create new classes.

The idea itself is not new and is also mentioned in (Vargas-Vera and Čeljuska, 2003a). However, the task is overly complex and brings a number of cases that need to be taken into consideration. Therefore, different fields of Artificial Intelligent are needed to be put together in order to achieve this goal.

The following passage only gives some general information in this field and in turn brings ideas to the foreground that should be considered in more detail.

Taking the idea into extreme, lets suppose that the ontology O contains only one class C with pre-defined slots $\{S_1, S_2, \ldots, S_n\}$. In addition, lets assume that the rules were learned and a new document *d* comes to be extracted. If appropriate entities are recognized within the document then the document can be clearly linked to the given class *C*. Moreover, the extracted entities can be used to construct an instance which will be fed into the class. At one point, the class *C* becomes over-populated with the number of instances and therefore the system might consider performing some kind of class management. The class management might be based on clustering. For this purpose, each of the documents linked to the class could be represented in a form appropriate for some kind of measure based on a bag-of-words, i.e. tf-idf (Salton and Buckley, 1998), in order to determine similarities between documents. If a few major clusters are discovered then those clusters might form new classes, which obviously will become subclasses of the class *C*. Because classes from an ontology contain defined slots, its therefore needed to determine what slots the new classes will contain. From the ontology definition it is clear that all subclasses inherit all the slots of its parents but in addition to these, they may also contain some specialized slots.

Furthermore, if an ontology consists of lets say three classes C_a, C_b and *C* where C_a and C_b are child of *C*, then the system has to distinguish between all of them when performing extraction. Thus the classification based on extracted entities is not sufficient to determine to which particular class the newly extracted instance should be fed into as they all share some common slots. The classification based on document similarity measure, i.e. based on tf-idf, is then essential in order to compute similarities between the document *d* and bag-of-word representations of all classes. Then the extracted instance for the document *d* would be fed into the class with the highest similarly to the document.

This only discussed one aspect that indeed has not been completely covered because as such it is overly complex. The other view could be performing clustering on the level of slots. It might happen that a class *C* contains two groups of instances. One with one half of filled predefined slots and two with the other half. Those two clusters/groups that would be discovered by clustering at the slot level could result into splitting class *C* into two more specific classes C_a and C_b.

At this point, one might see that the whole set of problems that raises when dealing with new class creation is huge and soon or later it will results into ontology class management and ontology refinement.

8 Validation of the entire system and experiments

8.1 Description of the set

For the experiments the ontology described in chapter 6.1 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 chapter 6.3, were gathered from five different archives including *KMi Planet*[33], *AKT Planet*[34], *Open University News*[35], *CISA Newsletter*[36] and *OU Business School News*[37]. All of them managed by KMi at the Open University. Thus they were not all in common format they were normalized by using a little Java program into one uniform form. In addition, entities such as names of institutions, organizations, people, places including buildings and cities were recognized in the set of documents. Based on the list of recognized entities a hand-crafted lexicon and semantical hierarchy were constructed for Crystal (chapter 6.6.1 presents a short part of those files). Even more, a list of all world countries, US states and their abbreviations was also added to the lexicon.

For the purpose of pre-processing a little Java program was written in order to normalized dates and insert periods after each sentence as described in chapter 6.5. All of the articles were annotated, as described in chapter 6.3, and every document was classified into one and only one of the mentioned class.

The table $8-1$ shows a number of documents for each of the class (*"Docs"*). In addition, it gives number of annotated entities (positive instances - *"Pos."*), number of

[33]KMi Planet - `http://kmi.open.ac.uk/news`

[34]AKT Planet - `http://news.kmi.open.ac.uk/rostra/news.php?r=8`

[35]Open University News - `https://intranet-gw.open.ac.uk/oulife/news` (password protected)

[36]CISA Newsletter - `http://news.kmi.open.ac.uk/cgi-bin/newsletters/cisa/cisa_archive.pl`

[37]OU Business School News - `http://news.kmi.open.ac.uk/rostra/news.php?r=16&t=1`

Table 8 – 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			

positive instance (*"Tot. pos."*) in total for a given class and total number of instances[38] for each class (*"Total"*). Slots that have not been annotated within the entire dataset are not listed.

8.2 Experiments

Two different experiments were performed:

- Validation of the entire system with the dataset

[38]one <ex>...</ex> is considered as one instance (chapter 6.5)

- Random trials to populate the Ontology

8.2.1 Validation of the system

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

- How using rule confidences effects the precision and recall.
- Which of the two ways of computing the rule confidence (chapter 6.7.1) 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 chapter 6.7.1, 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[39] 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:

[39] 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

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 (6.1) 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: $Merge = 2$.

Experiments with computation of class confidence C_{class} with both equations (6.7) and (6.8) were also run. However, since no significant change was observed only the (6.8) was used 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 $8-2$ 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[40] almost crosses 10% in case of *"Elimination"*, which implies that the computed average values is not statistically very reliable. The figures $8-8$ and $8-9$ gives better picture of the results.

It can be observed from the table $8-2$, 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

[40]difference between an average, minimum and maximum

Table 8 – 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

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 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[41] 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

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

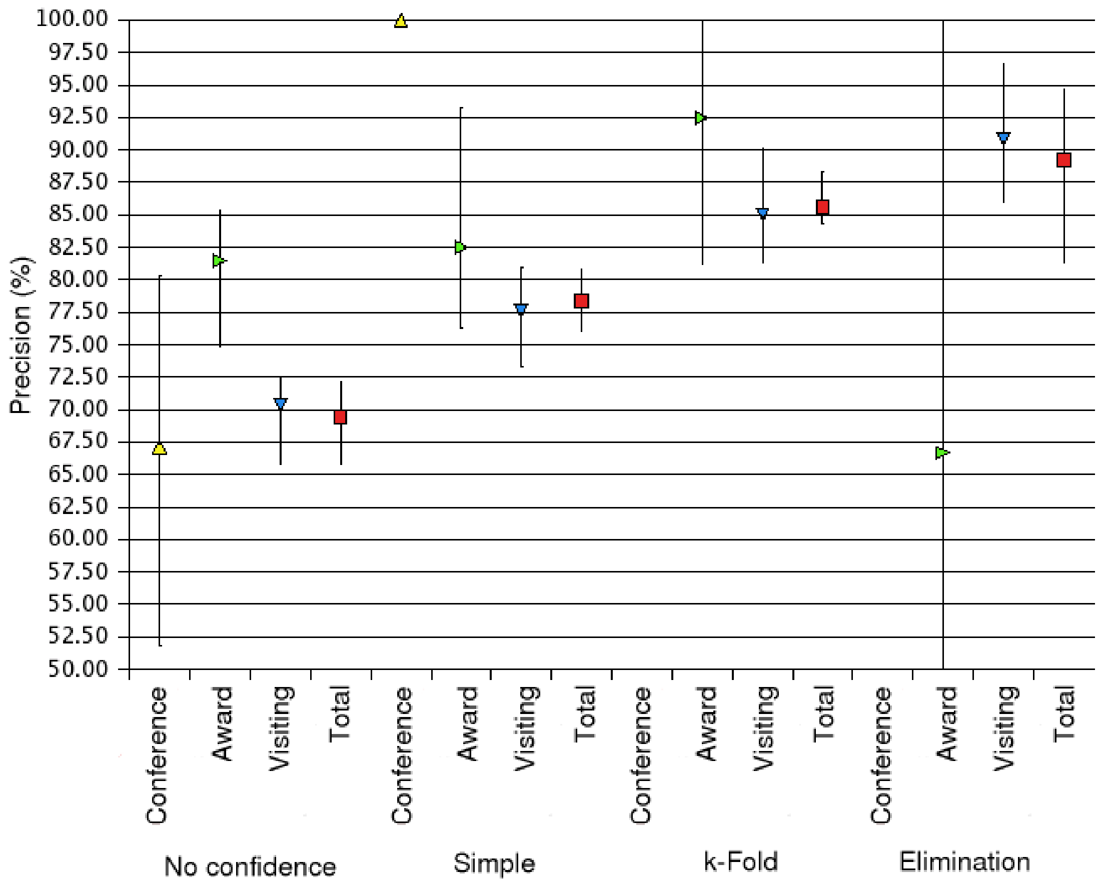


Figure 8 – 8 Precision for each method - per class and in total

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 8 – 9 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 8 – 1) was not very big. Thus the absolute number of positive instances is not very high. Learning on such a small set is

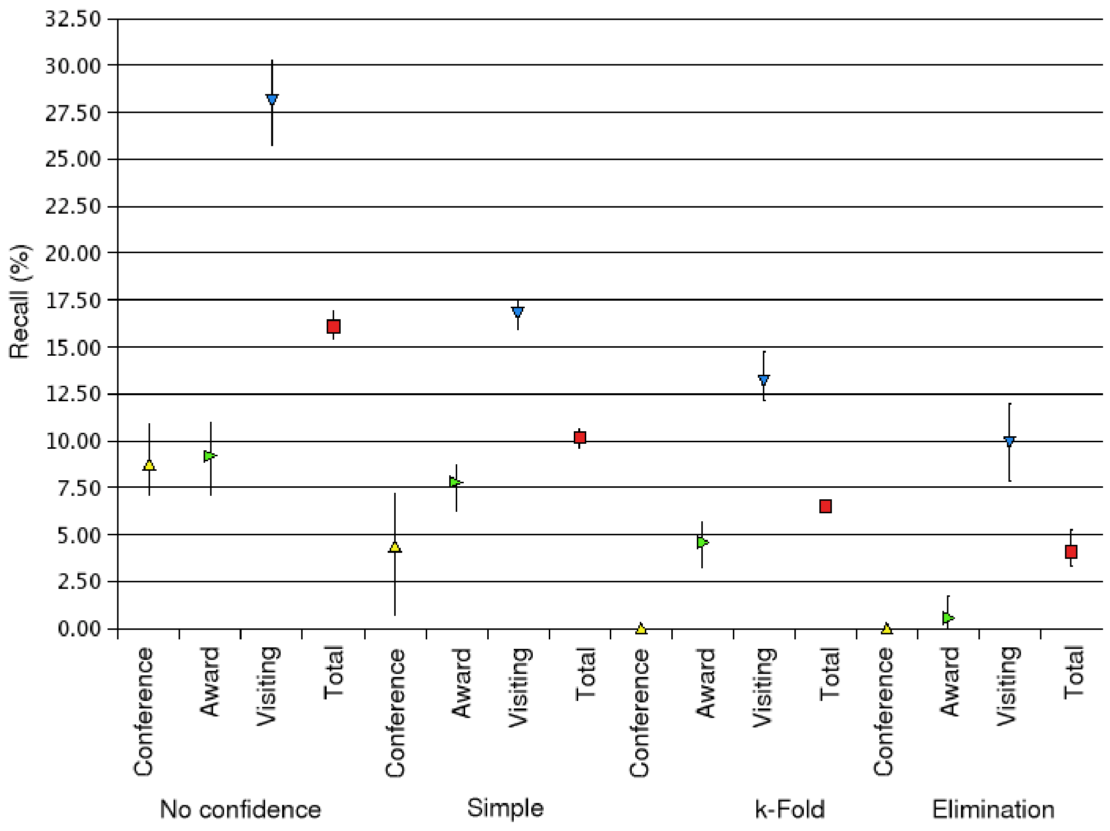


Figure 8 – 9 Recall for each method - per class and in total

always leading to more specific rules or overly generalized and causing variability to go high.

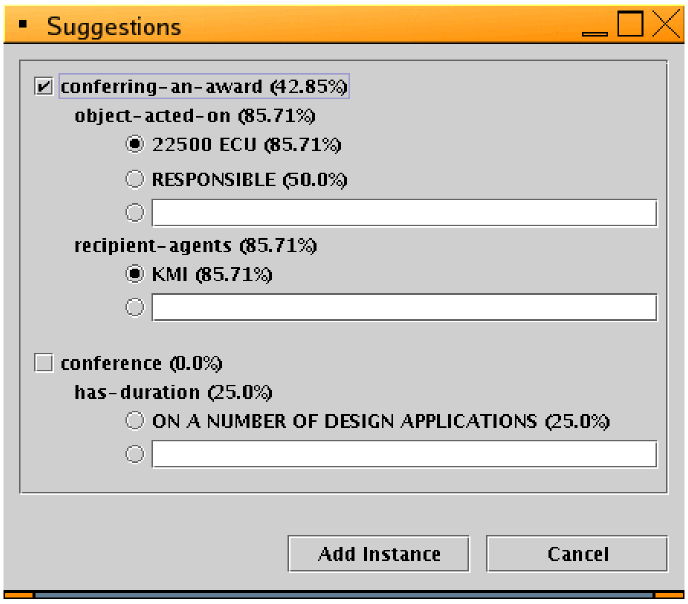
8.2.2 Trials to populate the ontology

The purpose of the trials was to see how the overall performance suited the user. From the entire set a few articles from each of the class were kept unseen for the learning. Then the system was taught on the rest of the set. Once the system had learnt, the extraction from the previously hidden articles was performed. Although, the system was not very successful to extract entities from class *conference*, it did very good with *visiting-a-place-or-people* or *conferring-an-award*. Sometimes the system extracted entities from more than one class but at each time it correctly identifies the right class and all its entities. A couple of screen-shots are provided to illustrate the extraction 8 – 10 and 8 – 11. A part of original text from which the suggestions were made is shown on the left side of each screen-shot and extracted entities were printed in **bold** within the text. The numbers in brackets next to each of either classes, slots or slot values in a window dialog represent confidence values at those level. Although looking at the original samples of text one can notice that not all of the entities were extracted, the extracted entities after applying threshold pruning were pre-selected correctly at each time.

Euro-Award for **KMI**

Enrico Motta 12-08-97

KMI has been awarded **22,500 Ecu** from the European Commission to carry out research in the area of knowledge-based systems. . . In particular, KMI will be **responsible** for defining the specification of the library of reusable components and for testing the approach on a **number of design applications**.

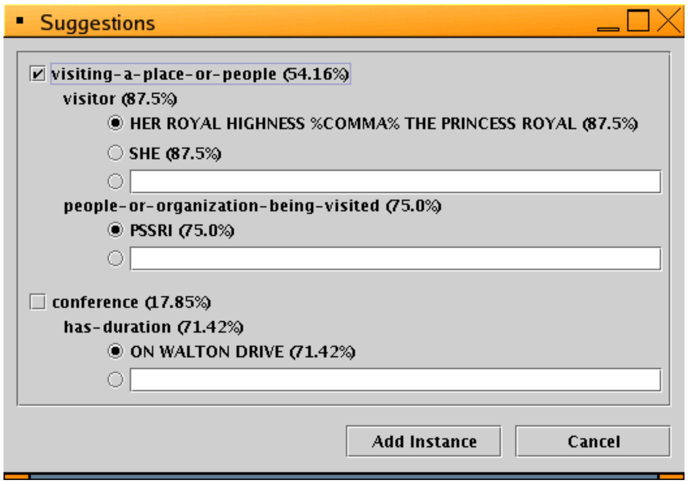


```
[✓ conferring-an-award (42.85%)]
    object-acted-on (85.71%)
        [✓ 22500 ECU (85.71%)]
        RESPONSIBLE (50.0%)
    recipient-agents (85.71%)
        [✓ KMI (85.71%)]
conference (0.0%)
    has-duration (25.0%)
        ON A NUMBER OF DESIGN APPLICATIONS (25.0%)
```

Figure 8 – 10 A part of original text and a window dialog with extracted suggestions

Royal Visit for Walton Hall

Her Royal Highness, The Princess Royal visits The Open University next Tuesday (February 10). . . During the afternoon, **she** will visit **PSSRI** to see the Hypervelocity Impact Laboratory. . . There will be no parking **on Walton Drive** north of the ring-road, and the Cellar Bar will be closed all day.



```
[✓ visiting-a-place-or-people (54.16%)]
    visitor (87.5%)
        [✓ HER ROYAL HIGHNESS %COMMA% THE PRINCESS ROYAL (87.5%)]
        SHE (87.5%)
    people-or-organization-being-visited (75.0%)
        [✓ PSSRI (75.0%)]
conference (17.85%)
    has-duration (71.42%)
        [✓ ON WALTON DRIVE (71.42%)]
```

Figure 8 – 11 A part of original text and a window dialog with extracted suggestions

9 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.

The thesis analyzed two major approaches in this area and Ontosophie, a system for semi-automatic population of ontologies with instance from text, was developed. Ontosophie 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 et al., 2001b; Vargas-Vera and Čeljuska, 2003a,b).

The experiments and the validation of the designed system showed that Ontosophie could be a valuable tool in the process of ontology population.

In addition to the population of ontologies with instances it would be useful to have a tool that is able to suggest the creation of new classes where appropriate. Therefore the thesis also analyzes this possibility. However, it concludes that this extension would require complex research, due to the fact that at the present time this area has remained predominately untouched.

Bibliography

DOMINGUE, J. – SCOTT, P. 2002. *KMi Planet: A Web Based News Server.* In proceedings of the Asia Pacific Computer Human Interaction Conference (APCHI-98).

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

ČELJUSKA, D. 2003. *Engineering of Ontologies from Text and Association Rules.* Preceedings of the 4th Slovak-Austrian Student Workshop on Data Analysis - WDA 2003, Malino Brdo, Ružomberok, Slovakia, ISBN 80-7097-523-7.

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).

ENGELS, R. H. P – LECH T. CH. *Generating Ontologies for the Semantic Web: OntoBuilder.* Toward the Semantic Web – Ontology-driven Knowledge Management, pages 91 – 115. ISBN 0470 84867 7.

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.

GRUBER, T. R. 1993. *A Translation Approach to Portable Ontology Specification.* Knowledge acquisition 5(2), pages 199 – 220.

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.

KHAN, L. – LOU, F. 2002. *Ontology Construction for Information Selection.* Proceedings of 14th IEEE International Conference on Tools with Artificial Intelligence, pp. 122-127, Washington DC.

MAEDCHE, A. – STAAB, S. 2000a. *Semi-automatic engineering of ontologies from texts.* Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering, SEKE2000, Chicago, IL, USA pages 231-239.

MAEDCHE, A. – STAAB, S. 2000b. *Mining Ontologies from Text.* EKAW 2000: 189-202.

MAEDCHE, A. – STAAB, S. 2000c. *Discovering Conceptual Relations from Text.* Technical Report 399, Institute AIFB, Karlsruhe University.

MAEDCHE, A. – STAAB, S. 2000d. *The Text-to-Onto Ontology Learning Environment.* Institute AIFB, Karlsruhe University.

MAEDCHE, A. – STAAB, S. 2001. *Ontology learning for the Semantic Web.* IEEE Intelligent Systems, 16(2).

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. 1996a. *An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains.* AI Journal, volume 85:101-134.

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

SALTON, G. – BUCKLEY, C. 1988. *Term weighting approaches in automatic text retrieval.* Information Processing and Management, vol. 24, no. 5, pp. 513 – 523.

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.

STRIKANT, R. – AGRAWAL, R. 1995. *Mining Generalized Association Rules.* In Proc. of VLDB’95, pages 407 – 419.

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

VARGAS-VERA, M. – DOMINGUE, J. – MOTTA, E. – BUCKINGHAM SHUM, S. – LANZONI, M. 2001b. *Knowledge Extraction by using an Ontology-based Annotation Tool.* In proceedings of the Workshop Knowledge Mark-up & Semantic Annotation, held in conjunction with the First International Conference on Knowledge Capture (K-CAP 2001), Victoria Canada, pages 5-12.

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.

VARGAS-VERA, M. – ČELJUSKA, D. 2003a. *Ontology-Driven Event Recognition on Stories*. KMI-TR-135, Knowledge Media Institute, The Open University.

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

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.

List of appendices

1. CD
2. User Guide
3. System Guide
4. List of figures
5. List of tables

List of Figures

List of Tables