ANNOTATING THE SEMANTIC WEB

By

Alexiei Dingli

THE UNIVERSITY OF SHEFFIELD

DEPARTMENT OF

DEPARTMENT OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Science for acceptance a thesis entitled "**Annotating the Semantic Web**" by **Alexiei Dingli** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dated: <u>July 2004</u>

External Examiner: _____
Professor Nigel Shadbolt

Research Supervisors: _____
Professor Yorick Wilks

_____
Professor Fabio Ciravegna

Examining Committee: _____
Dr Mark Hepple

ii

THE UNIVERSITY OF SHEFFIELD

Date: **July 2004**

Author: **Alexiei Dingli**

Title: **Annotating the Semantic Web**

Department: **Department of Computer Science**

Degree: **Ph.D.**   Convocation: **August**   Year: **2004**

Permission is herewith granted to the University of Sheffield to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

_____
Signature of Author

To my dear parents and girlfriend, who have always been the beam that supported me and to God who was always there to sustain that beam and me.

Table of Contents

Abstract

The web of today has evolved into a huge repository of rich Multimedia content for human consumption. The exponential growth of the web made it possible for information size to reach astronomical proportions; far more than a mere human can manage, causing the problem of information overload. Because of this, the creators of the web(10) spoke of using computer agents in order to process the large amounts of data. To do this, they planned to extend the current web to make it understandable by computer programs. This new web is being referred to as the Semantic Web. Given the huge size of the web, a collective effort is necessary to extend the web. For this to happen, tools easy enough for non-experts to use must be available.

This thesis first proposes a methodology which semi-automatically labels semantic entities in web pages. The methodology first requires a user to provide some initial examples. The tool then learns how to reproduce the user's examples and generalises over them by making use of Adaptive Information Extraction (AIE) techniques. When its level of performance is good enough when compared to the user, it then takes over the process and processes the remaining documents autonomously.

The second methodology goes a step further and attempts to gather semantically typed information from web pages automatically. It starts from the assumption that semantics are already available all over the web, and by making use of a number of freely available resources (like databases) combined with AIE techniques, it is possible to extract most information automatically.

These techniques will certainly not provide all the solutions for the problems

brought about with the advent of the Semantic Web. They are intended to provide a step forward towards making the Semantic Web a reality.

Acknowledgements

I would like to thank my supervisor Professor Yorick Wilks for giving me this opportunity and also for his support and comments through out these years. I would also like to thank Professor Fabio Ciravegna for the many times I disturbed him with questions, for the patience he has whenever I turn around and for the helpful comments he always gives me. Finally, but not least, I would like to thank Professor Niranjan for accepting to be the chair of my PhD panel.

Of course, I am grateful to my parents and girlfriend for their patience and *love*. Without them this work would never have come into existence (literally).

Chapter 1

Introduction

1.1 Motivation

Research on an experimental computer network driven by the fear of a nuclear war has become, many decades later, one of the most popular and important communication media in the world. It originally had the form of a wide area network referred to at the time as ARPANet (Advanced Research Projects Agency Network). Its purpose was to exchange military information efficiently in the eventuality that a nuclear attack would have made specific parts of the network unusable. Later on, universities recognised the power of such a network and started hooking themselves to it and also providing access to their students. Today this huge network has grown by many orders of magnitude, spans every corner of the earth and is more commonly referred to as the Internet.

In the past years, the web started growing at a very fast rate. The main catalyst for this increase in information[1] is the move (which happened some years ago) towards an information society succeeded by the present drive towards a knowledge[2] society. Globalisation is another factor that is bringing about huge information strains and demands on several organisations since organisations do not have to compete just

---

[1]Information is data which is interpreted.

[2]Knowledge is information transformed for effective use.

with other organisations in their own neighbourhood but they must compete with organisations all over the globe. For these to survive, in this fast changing world, it is imperative that they restructure themselves and move towards modern techniques of Knowledge Management (KM) in order to be able to handle all the knowledge available.

KM is an emerging, interdisciplinary business model dealing with all aspects of knowledge within the context of the firm. It seeks to improve the performance of organisations by helping users to capture, share and apply their collective knowledge with the aim of leading the organisation towards making optimal decisions in real time. These techniques make effective use and reuse of the knowledge within the organisation with the effect of increasing efficiency, productivity and service quality.

KM is not a separate process performed continuously by a particular department but it should happen in the background by everyone as part of the day-to-day running of the business. In this era where companies are moving towards installing in their company a Digital Nervous System(55), it is imperative that the information is available where, when and to whoever needs it, rather than hidden in the senior management offices. An effective digital nervous system allows a company's internal processes to operate smoothly and quickly, enables an organisation to respond to customer feedback quickly, gives it the ability to react to its competitive environment in a timely manner, and empowers employees with critical knowledge. The key is how effectively an organisation manages the flow of its digital information. All kinds of information-numbers, text, audio and video can now be put into digital form. Widely available hardware and software has also made it possible and necessary for organisations of all sizes to reshape the way they conduct their business. A combination of modern technology together with effective use of humans can make an explosive mix. Modern technologies can supply any kind of information at the touch of a button while humans are natural consumers of any type of information and have the unique

ability of efficiently and effectively processing the information.

The basic organisational unit of KM is the Community of Practice (COP)(16). A COP is a group of people, informally bound to one another through exposure to a common class of problems, common pursuit of solutions, and thereby themselves embodying a store of knowledge. Individuals are a very important asset in any organisation because they bring into the company an intellectual capital brought from their previous experiences. Also humans are capable of processing a wide variety of information represented in any kind of media. A COP is something around which all the information processes circulate. It becomes much more complex once the COP is spread amongst several distributed locations around the globe but this is inevitable since more users seem to be moving towards virtual collaboration.

For KM to be more effective, it can make use of Artificial Intelligence[3] (AI) techniques such as Knowledge bases to be able to filter and process the huge amount of data which users are faced with. Knowledge bases store the data in data structures which make it easy and fast for users to retrieve it back. These knowledge bases facilitate users who would like to perform from simple searches to complex queries. AI must face a number of challenges in order to make KM successful. It must work for a wide range of people, not just for the expert user. If complex tasks need to be performed by humans, AI techniques must disguise the burdens introduced by the complexity of the task. One of the main bottlenecks in any KM system (where AI can make a huge difference) is the extraction of information from various sources (See Chapter 3 for more details). Another bottleneck in terms of costs is the harvesting, organising and storing of information. Since KM cannot afford to use poor quality information which is inaccurate or out of date, KM techniques reuse information from multiple tasks and multiple audiences to reduce the costs and risks involved (See

---

[3]AI is the field concerned with making computer programs achieve goals in the real world using different techniques capable of learning and improving themselves.

Chapter 3.2.3). The possibilities are endless but unfortunately the main problem which is hindering further progress is the Internet itself.

Although the long tentacles of the web are reaching almost every aspect of our lives, it is still not mature enough and it is quite far from reaching its full potential. The committee of the W3C[4] have stated that:

> *for the Web to reach its full potential, it must evolve into a Semantic Web, providing a universally accessible platform that allows data to be shared and processed by automated tools as well as by people.*

The important question at this stage is what is the Semantic Web (SW)(10) and why is there a need for it? The SW is basically similar to the existing web but with an added dimension, the semantic dimension. This dimension does not visually alter the web pages in any way, and users will still see pages the same as they see them today. But this dimension adds meaning to web pages. Lets take a small example to understand better what we mean. Imagine we have a very simple Online Book Shop containing many pages with details regarding books. Let us also assume that every single page (which describes a book) must have at least the name of the book and the author of the book. When a person views such a page, he immediately identifies and distinguishes between the name of the book and the name of the authors. Humans manage to do so using a number of tricks. First of all they are equipped with a powerful data processing system commonly known as the brain. The brain is specialised at creating associations between this data and is capable of processing huge amounts of data. It is also very good at inferring new facts using existing facts combined with logical rules. Keeping this in mind, lets see how humans manage to identify the names of the authors. Since they are social animals, throughout their life, they come in contact with other humans like them, all with different names.

---

[4]http://www.w3c.org/

Therefore, it is easy for humans to distinguish between words and names because they have stored in their brain a partial database containing some of the names of the people they met through out their life. If an author's name is similar to the name of a person they met, the brain manages to generalise and infer that the particular word being analysed is probably the name of a person. Using this information and the conscious knowledge that the web site is an online book store, the person infers from his past experiences that a person mentioned in a web page of an online book store most probably is the author of a book. Since the authors were identified, (by exclusion) the other words present in the page must be the title of the book.

This was a very simple example. What happens if the author has a previously unseen name? Once again, the brain comes to the rescue. Using its powerful generalisation functions over the partial list of names which it has, it manages to extract a set of linguistic cues that offer hints towards identifying names. One such cue could be that names normally start with a capital letter. Another cue could point to the fact that the name of a book is normally called a *title* and therefore, the identification of the word *title* in the text could signify that the name of the book is to be found somewhere in its proximity. The number of cues and their complexity is enormous and humans use them continuously in order to understand the meaning of the world around them. But what would happen if a person had no experiences, no cues, nothing? It would be impossible to understand the world around him.

This is the current situation with any program that analyses data. When a computer program is created, it has no experiences, no partial lists, no memory, nothing, just instructions. So far these programs were created for closed domains where their mode of operation is well defined by their creator and any cues and lists of data they need to operate is also supplied by him. So if a program is needed to perform the same task as described above, a very simple approach would be to write a program that goes through the page, gets every word in the page and compares it with a list

of names. Obviously, the list of names must be supplied by the programmer. This is not very effective though since any list of names is finite and cannot contain all the names in the world. The programmer could also hardwire cues inside the program. A cue could be that a word starting with a capital letter is potentially the name of a person. But what happens if the word is the name of a company or it is the first word in a sentence? Those too start with a capital letter! These cases can be catered for by writing other rules, but obviously their complexity is always increasing!

A smarter approach would be to use AI techniques. AI is divided into several subfields and the subfield concerned with trying to understand human language is Natural Language Processing (NLP). The task of understanding human language is very complex. It involves identifying entities (such as names in a sentence), finding out the relationship between them, the context in which they were mentioned and a million other issues. For the simple task we mentioned before, we only need to use the techniques to identify names in a sentence. There are various, the technique most relevant is Named Entity Recognition and Classification (NERC) whose task is to identify names of people, organisations, currencies, dates and basically all other named entities. To achieve this task, it makes use of a set of rules and lists in a similar way to what we mentioned before. For a more specialised approach, we could make use of Information Extraction (IE). It refers to the activity of automatically extracting pre-specified information from a document. So in our previous example, an IE engine could be used to extract the names of the authors and the title of the book. The difference between a NERC and an IE engine is that normally a NERC is generic to any domain therefore it does not require any prior training before it is used in a new application. An IE engine is specific to a particular domain and requires training so that it is customised to the domain being analysed. The training phase normally involves either giving the IE engine some extraction rules or some example documents so that it can learn which facts it needs to extract. Obviously,

the training phase that makes use of an IE tool is more complex than a NERC since a NERC does not require training, but the NERC has some serious limitations. Let us go back to the scenario we were analysing before and complicate life a little bit more, by introducing the name of the editor in the page. So now we have 3 pieces of information on every page, the title of the book, the names of the authors and the name of the editor. A NERC would spot the names of the authors and that of the editor without any problems but it cannot distinguish who is who. I.e. it cannot tell that the first name found is the name of the author and the second name found is the name of the editor, or vice-versa. To perform this, we need to use an IE engine. The IE engine learns to spot linguistic cues from the document which help identify where the name of the author is normally found and where the name of the editor is normally found in the document. Both names are treated as two distinct entities because they come from different sets (the first from the set of people who are authors and the second from the set of people who are editors).

IE is a very powerful methodology, however most of the current IE technologies require skilled human effort to port the IE to new domains (e.g. NLP experts). To simplify their usage, the IE community uses techniques borrowed from another sub-field of AI called Machine Learning (ML). ML refers to a broad class of probabilistic and statistical methods for estimating dependencies between data and using these dependencies to make predictions. ML based IE is called Adaptive IE (AIE)(18; 7; 27). It can be exploited to allow naïve users (i.e. users knowledgeable about their domain but having limited knowledge when it comes to computing) to port IE systems so as to avoid referring to NLP experts. In AIE, only three items are required to extract information from any domain. First and foremost, the set of documents from where data needs to be extracted. Secondly, a set of training documents which are used to train the IE engine and thirdly an ontology[5] defining the concepts which need to be

---

[5]An ontology is a formal specification of a shared conceptualisation (61)(12).

extracted. Lets look at each one in more detail.

The set of documents used for extraction is normally referred to as the test corpus[6]. These documents can be free texts, structured texts (texts containing tables, lists etc) or a combination of both. Most important of all, a subset of these documents generally contain the information that needs to be extracted. In the scenario we are analysing, this corpus would consist of all the pages containing the names of the authors, editors and books.

The set of documents used for training is called the training corpus. It is made up of documents similar in structure and content to the test corpus but containing internal or external markers showing which information needs to be extracted. These markers are called annotations and a document having annotations is called an annotated document. In the case of outside annotations, the annotations are referred to as template annotations whereby a template is associated with one or several documents and its instances point to the elements found in the documents. The process of creating these annotations is referred to as the annotation process. There are several kind of markers which one can use, the most commonly used are called tags. A tag is a label assigned to a section of the document in order to identify some data. To understand better what we mean, imagine in the Online Book scenario we have the details regarding a random book as follows:

| Oliver Twist - Charles Dickens |

The text means nothing on its own for a machine. It is just a sequence of letters and symbols. Therefore, if we want the IE system to learn examples from this documents we must insert training tags inside the document to mark the examples as follows:

---

[6]A corpus is a large collection of writings of a specific kind or subject, used for linguistic analysis.

$$\langle BookTitle \rangle OliverTwist \langle /BookTitle \rangle - \langle Author \rangle CharlesDickens \langle /Author \rangle$$

The $\langle BookTitle \rangle$ tag marks the start of a book title while $\langle /BookTitle \rangle$ marks the end of the book title. The text in between those tags is the title of the book (it is referring to the book **Oliver Twist**). The same holds for the $\langle Author \rangle$ tag which marks the start and end of the author name (in this case **Charles Dickens**).

An ontology defines a subset of the real world made up of concepts and relationships between them. There exist several formats and standards for ontologies. A simple representation for the Book Shop scenario would be as follows:

$\langle Book \rangle$
$\rightarrow$ **has a** $\langle BookTitle \rangle$
$\rightarrow$ **is written by one or many** $\langle Author \rangle$
$\rightarrow$ **edited by an** $\langle Editor \rangle$

This representation does not follow any particular standard, its purpose is to illustrate the concepts behind an ontology. The words within $\langle \rangle$ are concepts while the words in bold represent the relationship between these concepts. Therefore, the above ontology states that a book has a title, it is written by one or more authors and is edited by an editor. An ontology is used to specify all the elements required in any domain and their relationship. It is useless annotating the document with tags that are not linked to an ontology because tags on their own are meaningless unless they are put in a context of some sort. If we look at the tag Editor on its own, we know that an editor can be:

1. A person who edits.

2. A person who writes editorials.

3. A device to edit film.

4. A program used to edit text.

If the context is not supplied, it is quite difficult to disambiguate what the particular word means.

So far we have seen that IE can be used to automate the process of adding annotations to any kind of page. These annotations are linked to an ontology and therefore have a well defined meaning. In other words, IE can be used to add meaning/semantics to pages. Unfortunately, this comes at a cost since it requires a certain amount of training. This training is normally supplied by a human whose job is to manually annotate the training corpus. This has been very successful so far since the tasks being tackled were restricted to closed domains. But when it comes to larger scenarios, the IE methodology starts facing problems. The largest scenario possible is without doubt the SW scenario.

In this scenario all the pages on the Internet need to be annotated with meanings so that they can be processed by programmes. The need for having these programs analysing the web arises from the fact that the web is full of both relevant and irrelevant information on any topic imaginable. The amount of information available is such that a human being cannot cope with it. This problem is also referred to as information overload. Therefore to make this process tractable, a human requires the aid of some automated agent(70) capable of analysing the data in order to search and filter out, redundant and irrelevant information whilst directing the user's search through this information space. The only problem with this approach is that the agents do not understand anything which is on the web because for them, all the information present is just a bunch of binary symbols. What the agents need is a way to help them understand the data so that they can process it in a more meaning full way. This is exactly the goal of the SW. Basically the SW is a silent revolution

whose aim is to add semantics to the web which can be understood and used by both humans and automated agents.

If we try to insert this meaning manually into web pages, we immediately realise that this task is not feasible. The documents which are already semantically annotated in the current web are few. The number of un-annotated web sites is enormous and, every day, more and more new web sites are being created. If we assume for a second that the process of manually annotating these web sites is tractable, and if we ignore that it is both time consuming and error prone when performed by humans, the task is still difficult to perform by non-experts. To help the users perform these tasks, there is lack of adequate tools capable of providing some kind of support to semi-automatically assist the annotation process.

IE can surely help in this respect but we are now faced with a scalability problem, i.e. how can we provide enough training documents to annotate the web? A possibility would be to make use of the data already available on the web to annotate the training documents. If we take a look back at the Online Book Shop scenario, all we require is a partial list of authors and book titles. Once we have this partial list, it can be used to automatically annotate some of the documents in the test corpus so that we create a training corpus. Then the training corpus is used to train the IE engine. The trained IE engine is applied over the remaining test corpus and new authors and titles are extracted. The annotation part is automatic because it is simply a matter of going through the authors and book titles in the lists, searching for them through the documents, and then adding the appropriate annotation around their instance. The lists can be found quite easily using any search engine and involve little effort on the human's side. The only problem remaining is the fact that these lists are normally distributed around the web probably stored in several databases. To overcome this we can make use of Information Integration (II) techniques. II refers to a collection of techniques which let applications access data as though it were a single database,

whether it is or not. It enables the integration of data and content sources to provide real-time read and write access. Therefore, if in the Online Book Shop scenario, we need to annotate some of the authors and we find two partial lists of authors, II methods can help us access this list as if it was a huge single list.

The combination of the different technologies mentioned above proposes a realistic path towards the creation of the SW. This marriage of AI with Information Technology on the web gave birth to a new field referred to as Web Intelligence (WI)(80)(106). WI is a very recent field that combines different research interests of individuals whose goals are related to producing next generation applications on the web. It seeks to exploit a combination of Artificial Intelligence (AI) and advanced Information Technology (IT) techniques on the Internet with the aim of creating Intelligent Web Information Systems (IWIS). WI seeks to build the next generation of Web-empowered systems, environments and activities in order to address a number of emerging challenges on the web.

Even though most of the techniques and methods used in WI have been in use for years (if not decades), the growth of the Internet created new needs in the research community. All the techniques used so far work fine in their particular restricted domain but most of them are not usable over the Internet due to the new properties and structure which this new medium has. The Internet brought with it several open issues which were not considered before. First of all, the structure itself is quite different than any other structure seen before. Not only, the documents themselves can be either free text, semi-structured texts or structured texts, but they also contain a number of multimedia objects within them. So existing theories and technologies need to be modified or enhanced in every field to handle these intra-document elements. But this is only a small portion of the problem, on the Internet one can also find inter-document elements by using hyperlinks. A side effect of using hyperlinks is that they extend the boundaries of a document beyond the "physical" boundaries

of that file. This obviously means that the complexity dealt with when analysing and processing these pages is much higher than with traditional documents. Apart from this, if we take a look at the growth rate of the internet(30), we immediately realise that the size of the net is almost doubling every year, therefore making it unmanageable for traditional systems. Basically, the web increases the availability and accessibility of information to a much larger community than other computer applications.

This new platform does not only bring about new problems, it provides some new and exciting possibilities.

Web Mining The process of finding correlations or patterns among dozens of fields in large relational databases (Traditionally known as Data Mining) is being extended to the web. The fresh stream of reliable information which Web Mining produces is vital for E-businesses, marketing departments and other decision making sections of an organisation since information can help them gain competitive advantage over their competitors. The use of explicit physical links between web documents (11) provides new logical connections between whole documents or segments of them. These techniques make it possible for the system to create personalised views of the web.

Web Information Retrieval Making use of information retrieval engines is the most popular way of finding information on the web. These search engines are and will be very important in the coming future but unfortunately they are currently very limited. A search through the statistics gathered by one of the most popular search engines available shows that the queries in English count to only about half of the searches [7] and other languages are slowly gaining popularity. This shows that there is a growing need for more multi-lingual

---

[7]http://www.google.com/press/zeitgeist.html

retrieval. There is also a need of more personalisation and therefore the creation of personal search agents. Finally, the SW is bringing about a new way of searching. There will be no more the need to invent a bag of words in order to try to find the information required. A search through the semantic space rather than through the syntactic space would make queries much more accurate.

Intelligent Web Agents The current generation of web agents are quite dumb. One of the main reasons is that they do not posses the necessary technology to process and understand what is written in web pages. That is one of the reasons why the SW is being setup. Once this is done, we will be able to see agents that can perform any kind of intelligent tasks from more powerful searches to execution of complex processes. This could also lead to the creation of virtual representations of individuals in the web. The tasks of these would be to execute tasks which they get from their counterpart in the real world.

In synthesis, WI seeks to address a number of future challenges. First of all a major goal is the SW per se. Once the SW is set up, it must be populated with agents. Agents can handle simple tasks easily but when it comes to complex tasks problems start arising since a degree of planning is essential. A plan is a sequence of actions used to achieve the goals, given an initial state. In this context planning information is interpreted from semantic Web documents while meaning and relationships of the words in the documents are specified in ontologies. If this scenario is reached, the next step would be to have agents distributed over the whole web and collaborating with each other. Finally, these agents will start representing humans in the real world, and they will start forming a social self-organising network. According to (80), if WI manages to tackle these challenges, we would have created what they call the Wisdom Web. This web would enable humans to gain practical wisdom of everything existing both in the virtual and real world, while providing a medium for sharing

knowledge and experience. It would also facilitate the creation of knowledge and social development by making use of its self-organising resources. WI is still in its infancy but it is a field that is trying to reach very ambitious targets.

So far, we only considered the technological revolution which the SW is expected to bring. But if the SW is to become a reality there is a need of a cultural change amongst the users. Individuals should be inclined towards a Knowledge-sharing culture where people are not afraid to share their knowledge and learn from others. Since in such a society, the most important resource is knowledge itself, incentives and recognition must be provided to encourage people to share their own knowledge. But people will not just share knowledge for the sake of it, first of all some users tend to be conservative, therefore, these knowledge technologies must infiltrate into day-to-day use by making gradual changes in the traditional systems. Secondly, if new fascinating technologies start emerging, one must make sure that they provide an added value to the user in real time even before the users start accessing it. The reason is that it is difficult to convince users to contribute to something which is empty, therefore, new technologies must not wait for users to bootstrap the process but they must seed the process before their technology is used. Users utilise a technology only if it solves their problems today and not someday! The basic organisational structure of the data is normally stored in knowledge structures or ontologies which allow the users to search in the "semantic neighbourhood" (29) of a concept, therefore making the search much more powerful and natural for the user. Since such an ontology is normally defined as a formal specification of a shared conceptualisation(60) and since every user might need to see a different conceptualisation, different ontologies can be used to present different views of the same data. This is very important since it allows future technologies to be transparent for the busy knowledge workers and only present to them what they need to see them.

The SW is such a huge initiative that it will take a lot of time and effort before it

becomes accessible and usable by everyone. In the coming chapters, we will present two methodologies that will contribute a small step forward towards reaching these targets and eventually making the SW a reality.

1.2 Goal

The future of the Internet may well be based around the SW concept. For this web to become a reality, most of the documents available online must be semantically annotated in order to be usable by automated agents. It is unrealistic to believe that people will just sit down and semantically annotate their pages for the sake of creating the SW. Many people still have no idea what the SW is and why it is being created. If we want the SW to be a success, this annotation process must itself be automated. The goal of this research will focus on how to use different approaches in order to automate as much as possible the process of semantically annotating the web. This task can be further subdivided into two distinct subgoals.

The first subgoal must cater for documents which are not represented adequately on the web and therefore, the need of user intervention is inevitable. Our research focuses on using a semi-automatic methodology combined with AIE techniques in order to help the users annotate documents. In this case AIE can be used in the traditional way i.e. by having a user annotate a set of examples and training a learning algorithm. To do this, we researched and examined how an annotation tool for AIE should work. We then propose a methodology that reduces the user's input as much as possible by exploiting the annotations which were already encountered in the document. So if the user annotates the name of a person which is repeated a further four times in the document being annotated, the system will cater for those annotations itself by annotating them automatically. Also, with the help of an AIE engine, it will be learning from the user what kind of annotations are required in the document. The system will continuously evaluate itself and when it learns to

reproduce these annotations, it will take over the annotation process and annotate the remaining documents for the user. The task of the user will change from the hard task of annotator, to the much lighter task of supervisor of the annotations inserted by the system. This methodology is expected to give two main benefits, first of all the effort expected from the user is diminished and focused only on the most difficult annotations. Secondly, the time taken to annotate will be reduced drastically.

On the other hand, the second subgoal is aimed to handle the huge amount of information available on the internet. Since the information found online is not centralised (But it is normally distributed amongst different sources such as lists, search engines, digital libraries etc.), it is common to have a lot of redundancy. This is given by the presence of multiple citations of the same fact in different documents. When known information is presented in different ways, it is possible to use multiple occurrences to bootstrap several recognisers which are then generalised further and utilised to recognise new information. Therefore the main focus of this research will be on how to exploit the redundancies of the web and combine them with AIE techniques in order to discover more information. The approaches we will explore are varied and range from using Human Language technologies such as Information Extraction to other information and resources available on the web. The result of this research will be a methodology capable of semantically annotating web pages using just some initial configuration from the user's side.

The path towards achieving these goals raised various interests amongst the scientific community. In fact throughout these past years, we managed to present our work to several different subgroups in the computing community World wide and also publish a long list of papers. The following, is the complete list of publications which were produced as a direct result of the research we conducted:

Book Chapters

- Ciravegna, Dingli, Wilks and Petrelli : "Using Adaptive Information Extraction for Effective Human-centred Document Annotation" in Text Mining (Springer Verlag 2003)

Papers

- Dingli : "Next Generation Annotation Interfaces for Adaptive Information Extraction " in 6 th Annual Computer Linguists UK Colloquium (CLUK03) , January 6-7, 2003 , Edinburgh, UK

- Dingli, Ciravegna and Wilks: "Automatic Semantic Annotation using Unsupervised Information Extraction and Integration" in International Conference on Knowledge Capture (K-CAP03) held in collaboration with the International Semantic Web Conference 2003 (ISWC03) Workshop on Knowledge Markup and Semantic Annotation (K-CAP03), Florida, United States, October, 20-23

- Ciravegna, Dingli, Iria and Wilks: "Multi-Strategy Definition of Annotation Services in Melita" in International Semantic Web Conference 2003 Workshop on Human Language Technology for the Semantic Web (ISWC03), Florida, United States, October, 20-23

- Ciravegna, Dingli, Guthrie and Wilks: "Integrating Information to Bootstrap Information Extraction from Web Sites" in IJCAI 2003 Workshop on Information Integration on the Web, workshop in conjunction with the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, August, 9-15

- Ciravegna, Dingli, Guthrie and Wilks: "Mining Web Sites Using Adaptive Information Extraction" in Research notes and demos section of the

10th Conference of the European Chapter of the Association of Computational Linguistics (EACL 2003), Budapest, Hungary, April, 12-17

- Ciravegna, Dingli, Petrelli and Wilks : "User-System Cooperation in Document Annotation based on Information Extraction" in 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), 1-4 October 2002 - Sigenza (Spain)

- Ciravegna , Dingli , Petrelli and Wilks : "Timely and Non-Intrusive Active Document Annotation via Adaptive Information Extraction" in Semantic Authoring, Annotation and Knowledge Markup (SAAKM 2002) , ECAI 2002 Workshop July 22-26, 2002 , Lyon, France

- Ciravegna, Chapman, Dingli and Wilks: "Learning to Harvest Information for the Semantic Web" in Proceedings of the 1st European Semantic Web Symposium, Heraklion, Greece, May 10-12, 2004

- Glaser, Alani, Carr, Chapman, Ciravegna, Dingli, Gibbins, Harris, Schraefel and Shadbolt: "CS AKTiveSpace: Building a Semantic Web Application" in Proceedings of the 1st European Semantic Web Symposium, Heraklion, Greece, May 10-12, 2004

Posters

- Dingli : "Active Document Annotation via Adaptive Information Extraction" in University of Sheffield, Department of Computer Science, Research Retreat 6 th November 2002, Sheffield *(Runner up for first prize)*

- Ciravegna, Dingli and Petrelli : "Active Document Enrichment using Adaptive Information Extraction from Text" in 1st International Semantic Web Conference (ISWC2002) , June 9-12th, 2002 Sardinia, Italia *(Runner up for first prize)*

- Ciravegna, Dingli and Petrelli : **"Document Annotation via Adaptive Information Extraction"** in The 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval August 11-15, 2002, in Tampere, Finland

- Ciravegna, Chapman and Dingli : **"Armadillo: Harvesting information for the Semantic Web"** in The 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval July 25-29, 2004, in Sheffield, United Kingdom

1.3 Thesis Structure

The first part of the thesis takes a look at the different technologies involved. Chapter 2 describes the problem of annotation for the SW together with a literature review. Chapter 3 describes the problem of automatically extracting and integrating semantic information. Both the IE and II sections are followed by a literature review.

In the second part, starting from Chapter 4 we will explain the semiautomatic methodology we propose. Chapter 4.4 will report the evaluation of the system and some concluding remarks. Then we take the problem a step further and explore the possibility of using fully automated techniques. In Chapter 5 we will explain the automated methodology we propose. Chapter 5.4 will report the evaluation of the system and some concluding remarks. Finally, Chapter 6 will present our vision of the future of annotation for the SW together with a general conclusion.

Chapter 2

Annotation

This chapter will deal with some of the most relevant research and work done in the use of Adaptive Information Extraction as support for annotation. The first section will highlight the main motivations behind our work. It will show some of the current problems in the field and the arising needs of the computing community. The following section will provide a comprehensive survey of the different approaches used so far to tackle this problem. It will also give an overview of similar systems used for other problems but which can provide ideas useful for our domain of research.

2.1 Motivation

The web of the future, most commonly referred to as the Semantic Web[1] is currently being constructed. It is invisible to the naked eye and most people who use the internet are unaware that the web is slowly changing not in style but in content. This change in content involves enriching the data contained in the web with semantic content in order to facilitate the work of automated agents trying to interpret the contents of a web page. A fundamental task in order to achieve this goal is to

---

[1]http://www.w3.org/2000/01/sw/

associate metadata with content. Metadata enriches the information on the web with properties about some given content even if the medium in which the information is stored does not directly support it. This kind of data is not only useful to describe content but it can also be used for organisational and classification purposes since it supplies additional properties which can be used for groupings.

The impact of this kind of data is unimaginable. If we just consider a common search problem as an example we can appreciate the potential of metadata. Until now whenever a user searches for information, this is done by selecting keywords which may be related to, or which are normally found in proximity of a particular concept. This has the obvious limitation that the search criteria is purely based on guessing related keywords based on world knowledge of the user. Because of this the search tends to be inexact and ambiguous. All this could be replaced by Intelligent agents capable of performing searches based on semantic information rather than on related keywords. These kind of searches would be more exact because the similarity of a page to the search query would not be based on bags of words and their occurrences in related contexts but rather on exact semantic information. This example can be extended to unlimited domains such as online shopping, any type of queries etc.

What is frustrating about the current web is that the technology to make this possible exists. Browsers became much more sophisticated than the original Mosaic[2] allowing customisable styles, applets, any kind of multimedia etc. Standards too have evolved from the powerful yet difficult-to-use SGML to much more usable XML and all its vocabularies like RDF, XHTML etc. Finally and most importantly all the information we need is available in the web pages, yet there is no automated way (so

---

[2]http://archive.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html

far) of extracting the semantic information from them. This is the reason why it is imperative to semantically annotate the web pages before they are of some use for the semantic web. Semantic annotation is the process of associating metadata to whole or parts of a documents.

But annotation is not the only limitation, having all the technologies and standards without having tools that make effective use of them is useless. A number of prototypical systems have been designed yet they still lack a number of fundamental features. The basic and most important feature lacking in most systems is the learning element. Manual annotation is without doubt a burden for human users because it is a repetitive time consuming task. It is a known fact that humans are not good at repetitive tasks and tend to be error prone. The systems that support some sort of learning do so in a batch mode whereby the learning is not managed by the application but rather by the user of the system. It can be seen clearly in tools such as MnM (38), S-Cream (63) etc whereby a user is first asked to annotate and then an IE engine is trained. There is a clear distinction between the tagging phase and training phase. This has the adverse effect of interrupting the user's work since the user has to manually invoke and wait for the learner in order to learn the new annotations. Apart from this, since the learning will be performed in an incremental way, the user will not be certain whether the learner is trained on enough examples considering the sparseness of the data normally dealt with. It may also be difficult for the user to decide at which stage the system should take over the annotation process, therefore making the handing over a trial and error process. Research towards making annotation semi-automatic or rather fully automatic in order to semantically annotate documents is still far from achieving this goal. Because of this there is still

lots of work necessary in this field of research.

2.2   Previous Work

This section highlights a number of existing systems that contributed in some way or another towards adding annotations to web text. Some of the systems propose to solve a different problem from the one we are trying to tackle. In our case the use of annotation is mainly restricted towards a particular group of people: those annotating documents in order to train an IE system. Most of the applications presented here tackle the task of generic annotation. I.e. they want to empower users by giving them the facility to add comments to a web site. The task of annotating for IE builds upon generic annotation but is a step further. It is not just a matter of adding comments in order to identify or highlight sections of the document but rather a learning process whereby the annotator is teaching the IE system what concepts are required to be learned. In our approach, the IE is constantly learning and when the IE system has seen enough examples, the annotations stop, since the IE system is capable of annotating the remaining documents. Therefore annotation is done in order to support an IE system and not just to annotate the documents.

Even though the programs in this section have a different scope from our task, they provide good indications as to how user annotation can be eased. This section will first take a quick look at the initial systems and then focus on the most relevant systems, those that can give a greater contribution to the solution of our problem.

1969 Standard Generalised Markup Language (SGML)[1] was developed by IBM in a research project whose task was to integrate law office information systems.

---

[1]ISO 8879:1986

It was meant to allow text editing, formatting, and information retrieval subsystems to share documents by having different meta information relevant to specific tasks, coexisting in the same document. SGML can be considered as the ancestor of modern markup languages.

1970-80's TeX[2] was one of the initial typesetting systems, developed by Professor Donald Knuth at Stanford University. The system is still used today with lots of enhancements like the addition of LaTeX[3] which allows the user to define higher level commands for common tasks. TeX reenforced the idea that layout information and content can be mixed in the same document which lie at the base of modern web languages like HTML.

1988 Xanadu[4] is the original hypertext project led by Ted Nelson. It is still alive and proposing ideas as to how hypertext should exists together with different styles of annotation. From it a number of prototypical applications were born like CosmicBook etc which seek to promote the ideas of markup.

1994 CoNote (56) is a computer supported cooperative work system designed to facilitate communication within a group via the use of shared annotations on a set of documents. The annotation system is used to study how people can collaborate when working with a set of shared documents. It allows a group of people to share a set of documents and to make comments about them, all of which are shared with the other members of the group. An annotation to a document is a comment on or question about the document. Annotations

---

[2]http://www-cs-faculty.stanford.edu/ knuth/
[3]http://www.latex-project.org/
[4]http://www.xanadu.com/

can also refer to other annotations. For example, they can answer questions or refute arguments.

1994 ComMentor (95) at the Stanford Digital Library was a research prototype for annotating web pages with a special customized browser. It required the installation of trusted, platform-specific client-side software. The browser provided a simple mechanism to identify a place on a currently visible page in which a comment could be written. In making an annotation, the user simply selects a region in the document being annotated, and the browser stores redundant information about it. When a page changes, an attempt is made to relocate the attachment point for the comment based on mechanisms such as embedded tags and/or string match. Users can browse for annotation sets, and put together their personal selection. The selected sets are stored as part of the user's profile for their browser, which is built from the information about the user on the annotation server. Among these sets, users can designate specific ones as "active". If a set is active, then the comments in this set for a document being retrieved are retrieved from the comment server for this set. Annotations can be indicated in an interface in a number of ways, including marginal markings (as in LaTeX), format-marking of annotated text (as in WWW browsers with underlined anchors), in-line presentation of the annotation text, and in-place annotation indicators. The tool is a generic meta viewer which can be used in a variety of contexts to get "preview" information faster than it is possible with a full document-view window. Since annotations are documents too, they can be recursively annotated.

1998 JotBot[5] is a Java applet that retrieves annotations from specialized servers and presents an interface for reading and composing annotations. It is a thin application when compared to all the others and only requires the installation of client-side software in a trusted Java virtual machine. The existing editor controls annotation size and does simple content validation. The existing version only supports small annotations due to the fact that it maintains lightweight IP connections. Although limited, this tool was the first to demonstrate lightweight annotations over the web.

1999 Third Voice was a commercial company that created a browser plug-in for annotation. The company is no more in existence mainly because of massive preassure against its products[6]. The system was a kind of newsgroup enabling people to talk and put comments in the contents of a site. The site was not altered in anyway but users could see the comments attached to it if they obtained the plugin.ThirdVoice was unpopular with many Web site owners, who were disturbed by the idea of people posting critical, OffTopic, or obscene material that would be presented on top of their site. Legal action was also discussed. Another issue was privacy since the annotations were centrally stored and controlled by Third Voice only.

1999 Annotate.net[7] is a commercial company which makes it possible for companies who are members of a privileged group to annotate specific pages in order to divert customer traffic to their own sites.

---

[5]http://www.icc3.com/ec/architecture/webannotations.html
[6]http://www.worldzone.net/internet/pixelsnttv/
[7]http://www.annotate.net

2000 The Annotation Engine[8] found at the Berkman Center for Internet and Society, Harvard Law School is an open-source server based proxy written in perl. The system makes use of a database in order to store user comments, pointers etc. These entities are then added to a document before it is previewed by a user. The original page is not physically added but what happens is that the page is loaded through a proxy which adds the annotations to it. This proxy approach allows the notes to be browser and operating system independent.

2002 VisualText™ is a development environment created by a commercial company called Text Analysis[9]. It integrates multiple strategies, including statistical, keyword, grammar-based, and pattern-based, as well as diverse information sources, including linguistic, conceptual, and domain knowledge, to quickly and efficiently develop text analysis applications. Apart from these it also implements methods for automatically generating rules from annotated text samples in order to aid the user during annotation.

2.2.1 The Alembic Workbench

The Alembic Workbench(34) is made up of a set of integrated tools that make use of several strategies in order to bootstrap the annotation process. The idea behind Alembic is that whenever a user starts annotating, the annotations being inserted are normally not bound to that specific document but can apply to other similar documents. In order to reduce the annotation burden as much as possible, the system should make use of such information. Therefore in Alembic, every piece of information which can be used to help the user is utilised. Eventually the task of the user changes

---

[8]http://cyber.law.harvard.edu/projects/annotate.html
[9]http://www.textanalysis.com/

from one of manual annotation to one of manual reviewing.

A user inserts annotations in Alembic by marking elements in the document using a mouse. Together with this method, some other strategies are used in order to facilitate annotation such as;

- Simple string matching was used to locate additional instances of marked entities.

- Has in built a rule language used to specify domain specific rules for tagging.

- Provides a tool that lists potential phrases in documents in order to help users identify common patterns

- Provide statistical information about words such as frequency count, potential importance etc.

The most innovative feature of this application is the use of pre-tagging. The main idea is that information which can be identified before the user starts tagging should be tagged in order to support the user by preventing him from wasting his time on elements which can be tagged automatically by the system.

Alembic also implements a bootstrapping strategy. In this approach, a user is asked to mark some initial examples as a seed for the whole process. These examples are sent to a learning algorithm that generates new examples and the cycle continues like this. Eventually a number of markings are obtained and are presented to the user for review. If the user notices that some of the rules are generating incorrect results, it is possible for the user to manually change the rules or their order so that the precision of the algorithm is increased. Although the machine learning rules generate quite good results, they lack two important factors which humans have i.e.

linguistic intuition and world knowledge. The Alembic methodology does not cater for redundant information, therefore allowing documents which are already covered by the IE system to be presented to the user for annotation. This makes the annotation process more tedious and time consuming for the user.

Experiments performed using the Alembic workbench has showed significant improvements in the annotation of documents. In several tests it was shown that users double their productivity rate. Also, with the data provided both by the users and automatically from the system, it was possible to train quite complex IE tools.

2.2.2 Annotation System

The Annotation System[10] developed at NCST Bangalore is one of the emerging annotation systems specifically targeted for the Semantic Web. This system works using a three tier approach made up of a thin client (which is normally a plug-in), an application server and a database used to store the annotations.

The system proposes an extension to a normal web browsers by attaching to it the functionality of annotating web pages directly from the browser. The user can select a particular text in the current web page and either annotates it or creates an annotation for the whole document. The system offers a restricted set of possible annotations which could be changes required, comments, corrections, examples, explanations and questions. This division into categories is important because it allows the user to query for annotations in a particular category. Apart from this, the annotations are grouped into subject hierarchies which a user can add/edit. He can browse through the hierarchy and select the appropriate subject under which his annotation can then be categorized. Annotations can also be stored in the corresponding subject

---

[10]http://www.ncb.ernet.in/groups/dake/annotate/

categories under different users and others can easily lookup the annotations made for particular subjects, once they come to know that a person X is doing research on topic Y. Once the annotation phase is completed the annotations are stored separate from the document either locally or on a remote server. A strength of this system is that virtually any resource on the web ranging from images, embedded objects, to anything with a URI (Uniform Resource Identifier[11]) can be annotated by this system. These annotations are limited to whole documents and not to parts of it. Another feature of the system is that users can not only view the annotations but also reply to any annotations they see. This facilitates the creation of discussions based on subparts of the whole document.

An important contribution of this approach is the fact that annotations are not stored within the document. This makes it possible for documents to have multiple annotations for different purposes. The filtering and searches performed on the annotation database makes it possible for the users to navigate through the huge annotation space without getting lost in the sea of annotations which one document could have. A user is capable of viewing/hiding annotations according to several criteria chosen by him.

2.2.3 Annotea

Annotea (74) is another system similar to the ones already described whose task is to associate metadata with content. The main difference between this and other systems is that Annotea is a W3C[12] initiative and although its main aim is to create a web-based shared annotation framework it does so using W3C standards as much

---

[11]http://www.w3.org/Addressing/
[12]http://www.w3.org/

as possible. The annotations are stored in an external document and the information is stored as an RDF (Resource Description Framework)[13] file. Together with RDF additional information is stored by useing XPointer[14], XLink[15] and HTTP[16]. The current implementation makes use of the Amaya[17] editor/browser and a generic RDF database. This kind of database was used because it allows other applications that use RDF to access it without having to change anything.

Annotea is capable of handling many types of annotations. It its simplest form, an annotation can be seen as a remark about a document referenced by a URI, made either by the document's author or by a third party. The annotations are stored in specialised servers and can be accessible and modifiable by anyone.

When Annotea was being designed, a number of issues were considered. It was built around open technologies and standards in order to simply its interoperability with other systems and to maximise its extensibility. To simplify the design of the system, annotations were limited to XML based documents and the annotated documents must follow the basic XML rule of well-formedness. The annotations used are typed using some sort of metadata for types. The types allow users to classify annotations. Annotations can be either private or public. Private annotations are stored locally while public ones are published on distributed annotations servers.

The system described is mainly a generic backend, but for the sake of our ends we will change focus towards the client side called Amaya. Amaya is a full-featured web browser and editor developed by W3C as a testbed. The system supports all the

---

[13]http://www.w3.org/RDF/

[14]http://www.w3.org/TR/xptr/

[15]http://www.w3.org/TR/xlink/

[16]http://www.w3.org/Protocols/

[17]http://www.w3.org/Amaya/

Annotea protocols and its main features include the creation, browsing and filtering of annotations. First of all a user can annotate three things: a whole document, a particular location in the document (based on where the cursor's position is) or the current selected text. As soon as an annotation is created, the user is asked to enter metadata describing the annotation. The browsing of the annotations occurs in two stages, first the annotations are downloaded either from the local repository or from remote servers and then they are merged together with the document. A user goes through the annotations by selecting them and their content plus all the associated metadata is immediately displayed in another window. Finally, since a document may be heavily commented, the system allows annotations to be filtered by author, by annotation type or by annotation server. Amaya also has a query language called Algae which allows the user to query the document for particular annotations only.

Annotea is a tool that allows users to associate metadata with web resources. The system is built on open standards therefore it can be usable by different applications. Also the system is virtually clientless since it can exist on top of any browser that handles the W3C standards. The model specified by this tool is used in various other annotation projects like Annozilla[18] which is basically an implementation for the Mozilla web browser. A further enhancement over Annozilla is the COHSE Annotator[19] which adds to the basic system the possibility of using a DAML+OIL ontology instead of just a list of concepts.

---

[18]http://annozilla.mozdev.org/screenshots/phoenix/annozilla_0.4
[19]http://cohse.semanticweb.org/

2.2.4 CritLink

The idea behind the CritLink tool[20] is based upon the original hypertext idea where users can insert comments in any web page in order to create an online discussion about the particular page. It also allows new types of links like bidirectional linking, extrinsic linking, fine-grained anchors, and link typing.

Bidirectional links are quite important especially for consistency's sake. Currently the web makes use of unidirectional links, this has the obvious limitation that whenever a link is followed, there is no simple way to trace the referent link. This omission makes the logical flow of any web page unclear and difficult to follow.

Intrinsic links refer to links found in a document. They are inserted by the author and no one else can modify or add new links. Extrinsic links are external links which are stored somewhere outside the particular document but which origin from a location within the particular document. This kind of link allows third parties to add relations to other documents.

The link typing allows a link to have attributes that describe the relationship between the two objects being linked. The reason for this is that whenever a link exists, there is no way of understanding what is the purpose of the link unless the link is followed or unless there is a description next to the link. This limitation makes it more difficult to determine whether a link is relevant or not.

Anchor Granularity define anchors that explicitly define the part of the document being anchored. Although this is part of the current HTML standard, most browsers simply ignore it. Whenever an anchor is clicked in a document, the user is taken to approximately the location referred by the anchor but there is no definite way of

---

[20]http://crit.org/ ping/ht98.html

knowing the exact area.

These limitations degrade the current user experience on the web. If a user notices an error on a web page, the page cannot be corrected or if a user wishes to add some references to an existing web page, it cannot be done. This leads to lots of incomplete information and waste of time. The web is not exploiting the benefits of having a community of users who can contribute to enrich the information found in web pages.

The CritLink software offers all these features with the added benefit that no change is required on the client's side. What happens is that before web pages are presented to the user, they are passed to an intermediate proxy called a mediator in order to modify the content with annotations. The mediator processes the document, examines the links in the database and inserts new links in the document in order to create reversible links. The system displays different icons depending on the type of the link. Therefore, the original page is not being altered in anyway and the users experience is being enhanced.

2.2.5   iMarkup

iMarkup[21] is a commercial tool whose purpose is to promote collaborative document creation. It is similar in spirit to the other tools already mentioned. The system makes use of a Context Review Server (CRS) in order to facilitate web-based collaboration between pre-defined groups of users.

The iMarkup tools allow users to annotate, organize and collaborate on Web pages and documents. The system allows the creation of various type of markup. This include:

sticky notes This note is like a Post-It Note for web pages. It is a symbol that

---

[21]http://www.imarkup.com/

can be "stuck" to any web page. The note is transparent therefore allowing text underneath it to remain visible. The sticky notes can also be minimized or displayed as a small icon on the Web page until the mouse is moved over it. At the end of the session, the note is saved.

freeform drawings The program comes with a Paint Brush tool that enables freeform drawing or to draw straight lines on any Web page. To create freeform marks on a Web page, the mouse is used to draw lines on the screen. The Paint Brush can be a variety of colors and sizes, even transparent.

text markup The system allows various text markup functions. Italics, underline, bold, strike-through and highlight are all supported. It also allows the use of a system similar to a highlighting pen whereby the background is changed to a bright color (highlight) and the text remains intact. Additional notes and comments can also be saved with the text markups.

voice annotations The latese version of the program also supports voice annotations and markups. The system works by simply adding your annotations by speaking "into the markups". Whenever a new voice annotation is inserted a small speaker is shown in the text exactly in the position where the annotation was inserted. To listen to a voice annotation, the user just needs to click on the speaker icon that is associated with the markup. When sharing markups the recipient can hear, instead of just read, ideas associated with the content on a Web page.

hotlinks and file attachments The markups also support file attachments and

hotlinks. This is useful when the user wants to attach additional support information to a document. Whenever a file is attached, a paperclip symbol appears exactly in the place where the attachment was inserted.

The key benefits of using this system are that the collaborative environment allows all team members to perform simultaneous reviews as part of the workflow process. The different kind of annotations allows a clearer delivery of ideas therefore making sure the message is sent across to the other members of the team.

2.2.6 MnM

MnM (38) is an annotation tool that aids the user whose task is to annotate documents by providing semi-automatic or fully automatic support for annotation. The tool has integrated in it both an ontology editor and a web browser. MnM support five main activities browse, markup, learn, test and extract.

Browsing is the activity of presenting to the user ontologies stored in different location through a unified frontend. The purpose of this activity is to allow the user to select concepts from the ontologies or whole ontologies which can be used to markup the documents in future stages. To do so, the application provides several views, various previews of the ontologies and the data stored in them. This part is also referred to as ontology browsing.

Markup is the activity of semantically annotating documents. This is done in the traditional way, i.e. by selecting concepts from the chosen ontology and marking up the related text in the current document. When a specific section is marked, an XML tag corresponding to a concept in the ontology is inserted in the body of the document.

For the learning phase, MnM has a simple interface through which several learning algorithms can be used. The IE engines tested were various ranging from BADGER[22] to recently Amilcare[23]. The role of IE in MnM is to learn mappings between annotations in the documents and concepts in the various ontologies.

In MnM there are basically two ways of performing tests, explicitly or implicitly. In the explicit approach, the user is asked to select a test corpus which is either stored locally or somewhere online and the system performs tests on that document. In the implicit approach, the user is still asked to select a corpus like the implicit approach. The difference is that the strategy for testing is handled by MnM and not all the documents are necessary used for testing. The corpus is divided between tests and training corpora automatically by the system.

The final phase is the extraction phase. After the IE algorithm is trained, it is used on a set of un-tagged documents in order to extract information. The information extracted is first verified by the user and then sent to the ontology server to populate the different ontologies.

MnM is part of a new breed of tools that integrate ontology editors with annotation interface. These together with the support of IE engines surely make the annotation task easier. Tools of the like of MnM are very important and will surely give a push towards the realisation of the semantic web.

2.2.7 S-CREAM

S-CREAM (63) (Semi-automatic CREAtion of Metadata) is an annotation framework that facilitates the creation of metadata and which can be trained for specific domains.

---

[22]http://www-nlp.cs.umass.edu/software/badger.html
[23]http://nlp.shef.ac.uk/amilcare/

On top of this framework, there is Ont-O-Mat, an annotation tool. This tool makes use of Amilcare, an adaptive information extraction engine. Amilcare is trained on test documents in order to learn information extraction rules. The IE engine is then used to support the users of Ont-O-Mat, therefore making the annotation process semi-automatic.

The origins of S-CREAM lie in another system called CREAM which is an annotation and authoring framework. The system makes use of an Ontology together with annotations. In this application, annotations are elements inserted in a document which can be of three types; tags part of the DAML+OIL[24] domain, attribute tags that specify the type of a particular element in a document also referred to as Metadata or a relationship tag also called Relational Metadata. A user can interact with CREAM in three ways; by changing the ontology and the fact templates manually, by marking concepts in the document and then associating them with elements of the ontology or by grabbing concepts from the ontology and marking them in the document available.

S-CREAM goes a step forward and exploits the power of adaptive IE offered by Amilcare in order to improve the CREAM model. Whereas before in CREAM, all the interaction occurred almost exclusively manually, the process in S-CREAM is as follows. The first kind of process is similar to the one in CREAM i.e. a user manually annotates and performs the mappings between the web documents and the ontology. The second kind is the same like the first but instead of a user, an IE engine (in this case Amilcare) is used to automatically perform the mappings. Obviously, this can only occur after the IE engine is trained on substantial number of examples

---

[24]http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218

provided by the user. The last kind of process uses a discourse representation to map from the tagged document to the ontology. This discourse representation is a very light implementation of the original theory. The reason being that discourse representation was never intended for semi-structured text but rather for free text. Therefore to overcome this limitation, the one used in S-CREAM is a light version made up of manually written logical rules in order to map the concepts from the document to the ontology.

S-CREAM is a comprehensive framework for creating metadata together with relations in order to semantically markup documents. The addition of an IE engine makes this process even easier and helps pave the way forward towards building automated systems in order to automatically semantically enrich the millions of documents which are still not ready for the next evolution of the web, the Semantic Web.

2.2.8   SHOE Knowledge Annotator

The Knowledge Annotator (82) (68) is a tool that makes it easy to add SHOE[25] specific knowledge to web pages by making selections and filling in forms. The tool has an interface that displays instances, ontologies and assertions. A variety of methods can be used to view the knowledge in the document. These include a view of the source HTML, a logical notation view, and a view that organises assertions by subject and describes them using simple English. The Annotator can open documents from both the local disk or the Web. These are parsed using the SHOE library, and any SHOE instances contained in the document are displayed in the Instances panel. The Knowledge Annotator can be used by inexperienced users to create simple SHOE markup for their pages. By guiding them through the process and prompting them

---

[25]SHOE is an HTML-based knowledge representation language.

with forms, it can create valid SHOE without the user having to know the underlying syntax. For these reasons, only a rudimentary understanding of SHOE is necessary to markup web pages. However, for large markup efforts, the Annotator can be quite slow.

2.2.9 SMORE

SMORE (3) is an integrated environment for creating online content. The system is made up of a **Semantic Markup** tool, an **Ontology** viewer/editor and an **RDF Editor**. The scope behind SMORE is to create a tool that facilitates the seamless insertion of semantic markup into any online content.

The idea being promoted in this tool is that anyone should be able to say anything about anything. To achieve this, there should not be constraining things like a predefined ontology. Because of this, SMORE tries to move away from the traditional knowledge engineering tools towards web authoring tools. Basically the insertion of semantics should be a side effect of the creation of a web site rather than the main concern of the user. This philosophy can be seen in several parts of the tool. First of all, it has a WYSIWIG[26] editor which allows users to insert annotations as RDF triples while they are creating the actual page. A user can also write emails from within the tool for which the standard fields (such as subject, to, from, body etc.) are attached to an email ontology and the user can annotate sub parts of an email. One typical use is whenever an email is sent requesting a meeting. Date, time, location, etc are annotated with concepts from an ontology which is sent as an attachment with the email which could be used by agents on the receiving side (if they are capable of processing the ontology). Some additional features of the tool include annotation

---

[26]What You See Is What You Get

of one whole or parts of an image, an advanced ontology management facility, web scraping and a semantic virtual portal (that provides other semantic links).

The tool basically promotes the idea of semantic annotation not as a process separate from other web processes but as something which can be integrated with any web application. It also removes the boundaries from traditional annotation and proposes annotation of various different types of media. A key feature is the flexibility offered by this tool whereby one or more ontologies can be used to model the domain and if one is not available, then the user is free to create his own ontology ad hoc.

2.2.10 The Gate Annotation Tool

GATE [27] (General Architecture for Text Engineering) from the Natural Language Processing Group at the University of Sheffield, is an infrastructure which facilitates the development and deploying of software components used mainly for natural language processing. The GATE package comes complete with a number of software components such as IE engines, Part of Speech taggers etc. One of the main features of the Graphical User Interface (GUI) provided with GATE is the annotation tool.

The annotation tool is first of all an advanced text viewer compliant with many standard formats. A document in GATE is made up of content, annotations and features (attributes related to the document). The annotations in GATE (as any other piece of information) is described in terms of an atribute/value map. The attribute is a textual description of the object while the value can represent any java object (ranging from a simple annotation to a whole java object). These annotations are typed and are considered by the system as directed acyclic graphs having a start and end position.

---

[27]http://gate.ac.uk/

The annotation interface works like similar tools whereby a user selects a concept from an ontology and highlights the instances of the concept in the document. The system also supplies some generic tools which are capable of extracting generic concepts from documents. These tools can be extended by use of a simple grammar to cover more domain specific concepts. Being an architecture, GATE allows other external components to be loaded which can aid to locate concepts. The results of these tools are then presented in the annotation interface in the form of a tree of concepts. The user simply needs to highlight a concept or a group of them and the annotations are immediately displayed in the document viewer as coloured highlights.

The GATE annotation tool is a powerful tool since it allows several independent and different components to work together. It also presents the user with a unified view of results obtained from these components.

2.2.11   Trellis Web

Trellis Web(57)(58) is an annotation tool with a different purpose. Instead of allowing users to add any kind of comment to a particular web site, it restricts these comments to a well defined set of annotations having specific meaning. These annotations are used to express either discourse relations, logical connectives or temporal relations. Since these relations can be too strict at times, some partial ones are also allowed. The reason why Trellis Web takes this approach is because its application is not oriented per se towards normal web annotation but rather towards information analysis and decision making. To make this analysis possible in an automated way, it is imperative that the grammar used is well defined and restricted. This makes it possible for the tool to perform some automatic inferences on the connected data available.

A plus of this tool is that since the comments inserted are based upon well defined

relations, the user does not need to justify or explain why a certain comment was written since it was inserted in a structured and logical way which can be interpreted unambiguously by anyone. If one of the decisions needs to be updated, this too does not pose a problem because all the explanation of why the decision was reviewed is available. Although this tool is very useful, it has an obvious drawback. Users must be accustomed to the vocabulary used to define the relationships. This can be limiting in some cases since the relations offered do not cover all the domains.

2.2.12   Yawas

An annotation tool that is oriented towards allowing users to insert comments directly in the web is Yawas. This tool as described in (36) tries to address a number of problems. Users do not remember why some pages were kept for future reference, they do not understand how they found a page (e.g. the navigational path is lost). The scope of Yawas is to have an annotation tool working with a Web browser in order to help to overcome problems such as handling the explosion of bookmarks in web browsers, remembering why bookmarks were created and storing information about context of page, e.g. the path that led to it.

Yawas helps the user solve these problems in a number of ways. First of all, by highlighting specific texts, the user can remember what was of interest in a particular page. Secondly, by keeping the parent URL of each highlighted document, the user is provided with a context of the book-marked pages. Thirdly, the annotations can improve personalised document clustering. Most of these features are not new and they all have a number of weaknesses. The main one being that each one of them has the potential of altering the contents of a page and it is very difficult to control who has the right to do so without being intrusive. The original solutions were oriented

towards a client server approach whereby the client is presented with a platform independent interface and the changes are stored somewhere remotely on a server. This approach was later discarded due to the fact that the delay imposed on the user in order to connect to the server would be annoying and intrusive in the annotation cycle. Apart from this, the server approaches raises some privacy issues about who has the right to store the data.

In synthesis, the Yawas system is a light annotation tool for the web. Annotations are stored locally due to the previously mentioned reasons and makes used of Dynamic HTML in order to dynamically modify the HTML document without the need of reloading. Yawas works by allowing the user to insert highlights in the HTML document simply by highlighting interesting areas. Comments can also be attached to any document. The comments inserted in the text can be considered as being a rough form of inserting semantics. In experiments performed with annotated documents using a classifier, the classifier produced more accurate and uniform semantic descriptions of page clusters when the pages were already annotated by different users than when un-annotated documents were used. These experiment suggests that annotations can improve the automatic clustering of Web pages.

The main problem with this application is that Yawas is platform dependent. Apart from this, it is a research project and there are already other commercial annotation tools that have appeared working on the same lines as Yawas and trying to steal its scene.

2.3 Conclusion

In this chapter, we saw a number of systems all with different features and characteristics. We started from the first systems that made a significant contribution to the Annotation task and gradually moved towards the different annotation systems we have today. To get a concise picture of all these systems, we have grouped these system using the following attributes:

Usability shows whether the program is difficult or easy to use. This feature was quite crucial in some of the systems mentioned above since a difficult to use interface sometimes resulted in the silent death of the system.

Features refers to those characteristics that distinguish a system from the rest. E.g. some of these programs mentioned before have special features that avoid duplicated annotations. Such features are too varied to be listed all, but with this attribute, we are distinguishing between those systems that give an added benefit to the user and those that just present a plain annotation tool.

Type refers to the different kind of applications and how they work. These can be subdivided into four groups:

Stand Alone applications work on a single machine and they are not connected to a server somewhere else. The benefit of this approach is that the system generally works faster since no remote invocations occur. The downside of the system is that since the program is self contained, it does not share information with others and thus there's no collaboration involved.

Client-Server applications typically have a thin client on the user's machine which accesses a remote server. The benefit of such a system is that since these programs require a lot of processing power to function correctly, all the processing is done remotely on the server and only the results are sent to the thin client. Thus, the user does not need a powerful machine to use the system.

Proxy based applications are similar to Client-Server systems but instead of using a thin client, they typically use a normal web browser to access the annotated page. In fact, the page is not annotated on the client machine but rather when it is being passed through the proxy. This has the same advantages of a Client-Server approach with the added benefit that the user does not have to install any client application.

Distributed applications can be similar to either Client-Server or Proxy based approaches but since the system is distributed, the same data can be accessed and modified by different users. Therefore, such systems would have additional features that handle different user rights thus allowing secure collaboration between different users.

IE distinguishes between systems that make use of IE and those that don't. It also differentiate between those systems that make use of a Batch approach or an Incremental approach (See section 3.2.1).

Embedded/Linked refers to the method used to store annotated documents. The former method modifies the body of the document and stores the annotations inside it. The latter does not alter the document containing the data in any

way but stores the annotations in a separate document which contains links to the information inside data document.

Annotations distinguishes between the different kind of annotations which can be stored inside a document:

Layout annotations are used to define how a document should be displayed visually, such as **Bold**, *Italics*, etc.

Information annotations are used to insert any kind of information, such as comments, inside the document. They are generally free text which are not associated with a particular topic, therefore their purpose is generic and not well defined.

Semantic annotations are used to assign meaning to a portion of text. These annotations are normally associated with an Ontology and have a well defined meaning.

Public attribute distinguishes between systems developed by public entities and those developed by commercial organisations.

Secure systems offer a certain degree of protection when a user annotates a document. The security features are various and include protecting the document from unauthorised modifications or viewing of annotations, protecting document editing in distributed environments, etc.

Systems	Usability	Features	Type	IE	Embedded/Linked	Annotations	Public	Secure
SGML	Difficult	N/A	N/A	N/A	Embedded	All	Yes	No
TEX	Simple	N/A	N/A	N/A	Embedded	Layout	Yes	No
Xanadu	Simple	N/A	N/A	N/A	Linked	Information	Yes	No
CoNote	Simple	Yes	Distributed	N/A	Linked	Information	Yes	No
ComMentor	Simple	Yes	Client-Server	N/A	Linked	Layout/Information	Yes	No
JotBot	Simple	Yes	Client-Server	N/A	Linked	Layout/Information	Yes	No
Third Voice	Simple	Yes	Client-Server	N/A	Linked	Layout/Information	No	No
Annotate.net	Simple	Yes	Client-Server	N/A	Linked	Layout/Information	No	Yes
Annotation Engine	Simple	Yes	Proxy	N/A	Linked	Layout/Information	No	No
VisualText	Simple	Yes	Stand Alone	Batch	Embedded	Information	No	Yes
Alembic Workbench	Simple	Yes	Stand Alone	Batch	Embedded	Information	Yes	Yes
Annotation System	Simple	Yes	Client-Server	N/A	Linked	Semantic	Yes	No
Annotea	Simple	Yes	Client-Server	N/A	Linked	Semantic	Yes	No
CritLink	Simple	Yes	Proxy	N/A	Linked	Information	Yes	No
iMarkup	Simple	Yes	Distributed	N/A	Linked	Information	Yes	Yes
MnM	Simple	Yes	Stand Alone	Batch	Embedded	Semantic	Yes	Yes
S-Cream	Simple	Yes	Stand Alone	Batch	Embedded	Semantic	Yes	Yes
Shoe Annotator	Simple	No	Stand Alone	N/A	Embedded	Semantic	Yes	Yes
SMORE	Simple	Yes	Stand Alone	N/A	Embedded	Semantic	Yes	Yes
Gate Annotator	Simple	Yes	Stand Alone	Batch	Linked	Semantic	Yes	Yes
Trellis Web	Difficult	Yes	Client-Server	N/A	Linked	Information	Yes	Yes
Yawas	Simple	No	Stand Alone	N/A	Linked	Information	Yes	Yes

Table 2.1: Comparison of the different Annotation systems

Table 2.1 lists all the different systems mentioned earlier and their attributes. From it, we can identify a number of desirable features which an annotation system should have. First and foremost, such a system should be simple to use. Simplicity does not imply that our system will lack special features. It just means that the system should still have powerful features which enhances the user's experience, but they should be easy to use. The type of the application should be considered carefully because a stand alone system would discriminate users having a slow machine whilst the other approaches would discriminate users who do not have access to a network. Because of this, an ideal system would cater for all the different situations. When it comes to the issue of using IE, the most desirable feature is having a system capable of interactive IE rather than batch IE. In fact no system mentioned above is capable of having an interactive IE session with the user. The issue of having embedded or linked annotations is very debatable, there are many pros and cons to both approaches and it is very much dependent on the IE engine used. Regarding these issues, in Chapter 3 we will take a look at the various IE systems available.

The annotations which such a system should cater for are Semantic annotations. There are two reasons for this, first of all the scope of the Thesis is the SW and secondly, Semantic annotations can be considered as being a superset of Information and Layout annotations since both of them can be expressed as Semantic annotations. Finally, the program must be secure enough especially if it handles distributed domains. These features and many others will be demonstrated in Melita, the annotation tool mentioned in Chapter 4. The following chapters will then keep on building on the ideas implemented in that tool.

Chapter 3

Information Extraction and Integration

The two main technologies which are most relevant to our work are without doubt Information Extraction (IE) and Information Integration (II). IE provides the adequate capabilities necessary to learn new concepts, while II allows our systems to handle huge amounts of information harvested from several different sources. In Section 3.1, we will first take a look at why we need IE in the first place. To do so, we will focus on a specific subgroup of IE techniques most commonly referred to as Adaptive Information Extraction (AIE) techniques. This section will highlight the main motivations behind our work. It will show some of the current deficiencies in the field and the arising needs of the computing community. Section 3.2 will start with a brief introduction in order to explain some technical concepts used in the IE field and will continue towards giving a comprehensive survey of the different approaches used so far.

Section 3.3 will take a look at II and will try to give an overview of the most relevant research done so far. It will highlight the main motivations behind our work and will illustrate some of the main problems in the field. Section 3.4 will then give

51

an overview of the different approaches developed so far. Unfortunately, due to the fragmented nature of the field we will only scrape the tip of the iceberg in our reviews since a more comprehensive review of all the topics found in the II field would be beyond the scope of this thesis. Instead, we will give an overview of the major topics found in the field.

3.1 Motivation for using AIE

AIE is quite a recent field, it has its roots in traditional Information Extraction (IE) but differs in a subtle aspect. Whereas traditional IE is more concerned with producing systems which are capable of extracting information, AIE still has that as its main goal but considers equally important the usability and accessibility of the system. Although traditional IE managed to find the required information with a high degree of success, it did not succeed much whenever the need arose to port the existing system to new domains easily. There are a number of reasons for this. Such systems are normally complex with large amounts of domain dependent data, because of this, to port such systems, an IE expert is required. Also, due to the complexity of such systems the process to port them to a new domain is very time consuming. In synthesis one of the main barriers to the use of IE is the difficulty in adapting IE systems to a new scenario (105). AIE seeks to overcome this problem by creating systems which are adaptable to new domains without the need of much human input. This is achieved by exploiting the power of Machine Learning (ML) in order to learn how to automatically tune the application to new domains. This field is still in its infancy and the future will surely hold much more powerful tools than we have today.

In Adaptive IE a typical life-cycle is composed of scenario design, system adaptation, results validation and application delivery (23). The adaptation part which is maybe the most problematic of all the elements in the life cycle can be further subdivided into four main tasks. These are

- Adapting to the new domain information.

- Adapting to different sub-languages features.

- Adapting to different text genres.

- Adapting to different types.

The adaptation problem has been around for quite some time but was boosted in the last couple of decades due to the fact that the IE movement has grown by exploiting and joining the recent trend towards a more empirical and text based computational linguistics (105). The proliferation of the Information super highway is also blowing this problem to new heights since document content is evolving from old text based documents to a richer document model full of multimedia elements where the sequential structure is being disrupted due to the use of hyperlinks. Therefore, the text based methods alone, which were normally used, are not much capable of adapting to this new model.

The first approaches in the territory of AIE all made use of shallow approaches. The main reason being that it was not known how to train complex approaches using only knowledge of domain and annotation. Since shallow approaches are less complex, they are are also much simpler to port to new domains than deeper ones. Also, deeper approaches rely heavily on domain dependent data therefore making the task of adapting them to new domains much more difficult. Although the shallow

approaches can extract a wide range of information, they are still not capable of handling more complex tasks whereby different relations are involved. For these tasks, deeper approaches which make use of semantics are a necessity. Due to the fact that AIE can be easily ported to new domains, it can also be used to set up dynamic scenarios automatically. This will be very useful when harvesting knowledge from the web since domain information my not be known beforehand or might be incomplete and therefore an AIE system must be set up on the fly. Research in this area is still in its infancy and that is why we think there is the need for more work.

3.2 Previous Work in AIE

This section will give a brief overview of the different types of IE systems available. It will give a flavour of what we believe are the most relevant systems to our work. Before doing so, we will introduce some terminology highlighting the important differences between the existing systems.

3.2.1 Information Extraction Terminology

Shallow vs Deep approaches

IE systems fall into one of two major categories, those that make use of shallow NLP and those that use deep NLP techniques. Shallow approaches (See (77; 46; 32; 24; 102; 94; 19), etc.) are mainly concerned with the syntax of a document and make use of very little semantic information. They typically include tokenisation, part-of-speech tagging, parsing and produce a regularised form which may be anything from a partially filled template to a full logical form. The deeper approaches (See (53; 72; 54; 50), etc.) also makes use of the information obtained from the syntactic analysis but they combine it with additional external knowledge of the domain such

as heuristics. They attempt to integrate the information from the individual sentence representations with common world knowledge into a larger structure. Ultimately this structure is either the final template or serves to provide, the information for that template after it is analysed using deeper techniques. These techniques generally make use of logic to validate the semantics of the sentence and to deduce new information. The deeper approaches can be considered as being a continuum of the shallower approaches.

Single vs Multi Slot

The different systems can learn either a single concept or multiple concepts. These are normally refereed to as single and multi slot. The reason why this terminology is used, is because traditionally, the task of IE was to fill slots in templates. A single slot handles simple atomic concepts without any relations between the concepts. Normally the relations are handled at a higher level mainly by using an ontology. Also the concepts in this kind of domain are unique and can be identified fairly easily. This task is normally tackled quite easily by the shallow NLP approaches. In multi slots, concepts are complex entities. They are normally made up of several sub-concepts and a document may contain several similar concepts which are irrelevant for the IE task at hand.

Batch vs Incremental Learners

Learning systems either learn in batch or in incremental mode. Most systems operated in batch mode. This means that the examples are collected and the system is trained on all the training set. There are a number of reasons why this approach is preferred. First of all, it is much simpler to invoke a training session on all the set rather than

on individual examples. Secondly, the amount of processing power required for every training phase is very expensive therefore, invoking such a phase often would result in degrading the performance of the system. Third, in order to speed up the process the examples are normally ordered in an efficient way so that whenever paths in the search space are discarded, this would discard all the paths emerging from them. If the other approach was used, this would be impossible to perform. The incremental approach works by adding rules to the test set one by one and evaluating each time the effect of that rule. This has some obvious benefits, basically rules are evaluated according to individual merit and not as part of some collection. Therefore using this approach, the learner can be able to evaluate the coverage of individual rules learnt and using such evaluation it can decide when to stop the training session. This obviously makes the task more complex and might take longer to converge since in this approach, rules that produce local maxima are removed in order to find global maxima.

Interactive vs Non-Interactive Learners

Interactive learners make use of an Oracle in order to verify and validate their results. These Oracles are small programs created beforehand which say whether a concept is part of the domain we are searching in or not. Interactive learners will be explained further in Chapter 5. Non-interactive learners do not use such programs and normally, their only source of feedback is from the training provided by the user. Both approaches have their strengths and weaknesses, but the non-interactive approach is more oriented towards AIE since it requires no additional settings apart from the information already supplied by the user.

Top-Down vs Bottom-Up learning

An issue to consider is whether the system starts learning from scratch or from a hypotheses. The former is referred to as a top-down approach. What happens is that the learner starts from a very generic rule and gradually adds cases to constrain the rule. It then evaluates the validity of the new rule. The cycle continues until a good rule is obtained. The latter (also referred to as theory revisers and bottom-up approaches) begin from an initial hypotheses which is revised as the learning proceeds. This hypotheses normally starts with all the features available and systematically start dropping some of them (it relaxes the constraints of the rule). At every stage, the new hypotheses is tested and the one that gives the best results is retained.

Text types

A positive outcome of the second World War was without doubt the rapid development of computers. These huge machines were capable of processing large amounts of text much quicker than any human could possibly do. During those days, text types were not an issue because the only kind of text available was free text. Free text contains no formatting or any other information. It is normally made up of grammatical sentences and examples of free texts range from news articles, quotes, stories etc. The text contains two types of features, syntactic and semantic features. Syntactic features can be extracted from the document using tools like part-of-speech taggers, chunkers, parsers etc. Semantic information can be extracted by using semantic classes, named entity recognisers etc.

In the 60's when computers were being used in businesses, and information was being stored in data bases, structured data became very common. This is similar

to free text but the logical and formatting layout of the information is predefined according to some template. This kind of text is quite limiting in itself. The layout used is only understandable by the machine for which it was created (or other compatible ones). Other machines are not capable of making sense of it unless there is some sort of translator program. Syntactic tools are not very good at handling such texts. The reason being that most tools are trained on free texts. Apart from this, the basic structures of free text (such as sentences, phrases etc.) do not necessary exist in structured text. Structured text mainly has an entity of information as its atomic element. The entity has some sort of semantics according to its position in the structure. Humans on the other hand show an unprecedented skill of inducing the meaning most of the time. Yet, they still preferred to write using free text. Therefore, these two types co-exist in parallel.

With the creation of the internet another text type gained popularity, semi-structured text. Unfortunately it did so for the wrong reasons. Before the semi-structured text era, information and layout were generally two distinct objects. In order to enable easy creation of rich document content on the internet, a new content description language was created. The layout features which until recently were hidden by the applications became accessible to the users. The new document contained both layout and content in the same layer. This made accessible to the user new structure elements such as tables, lists etc. Now users could create documents having all the flexibility of free text with the possibility of using structures to make clear difficult concepts. Obviously this complexity made it more difficult for automated systems to process such documents. NLP tools worked well on the free text part

but produced unreliable results whenever they reached the structured part. Standard templates did not exist for the structured part because the number of possible combinations of the different structures is practically infinite.

Generalisation

Some other issues worth explaining at this stage have to do with the actual implement strategies of adaptive algorithms. First of all it is worth clarifying that these algorithms although they are concerned with a traditional NLP task are a hybrid between NLP and ML approaches. Therefore the problems which must be faced when designing such algorithms are inherited from both fields. A critical issue in developing a learning algorithm is generalization. Basically, it is the issue of how to produce generic rules out of domain specific data. There are a number of ways to do so, such as using regular expressions etc. In the systems below we will describe several approaches which accomplishes this task.

Overfitting vs Underfitting

The issue of generalisation is quite important because if it is tackled badly, the system may suffer from what is normally referred to as underfitting or overfitting. A learner that is not sufficiently complex can fail to detect fully the model in a complicated data set, leading to underfitting. On the other hand, a learner that is too complex may fit the noise, not just the model, leading to overfitting. Overfitting is especially dangerous because it can easily lead to predictions that are far beyond the range of the training data and can also produce wild predictions even with noise-free data. Underfitting produces excessive bias in the outputs, whereas overfitting produces excessive variance. But this phenomena does not only occur due to the generalization strategies

of the learner, it is also caused by Data sparseness. This problem is concerned mainly with how representative the training data is of the concept we are trying to learn. Most of the data normally available is incomplete or inconsistent, the effect of this is that the model learnt by the learner is partial and does not represent adequately the target concept. Therefore our algorithms must be capable of creating a model as representative as possible of the target concept using the noisy information available. A number of approaches tackle this issue such as the use of linguistic information in order to perform generalization, the use of as much data as possible (which although desirable not always possible) or the use of cross-validation in order to cross check the expressibility of the rules, between subsets of the training data set.

3.2.2 Shallow Approaches

Wrapper Induction

With the rapid expansion of the Internet and before the advent of the Semantic Web, data was being encapsulated in documents containing lots of fancy layouts in order to make the data more appealing to the user. Although this made things easier from the user's point of view, it made it extremely difficult for the automated agents to extract information from such documents since they did not know how to handle the composition of layout and content. Since the layout information is normally embedded in the document, the traditional Natural Language Processing tools like Part of Speech taggers, parsers etc produce lots of erroneous results because the documents are not just pure text but also contain the layout elements. Because of this, IE was performed by manually constructing wrappers in order to define where the data is located. The wrappers model the data and layout surrounding the information which needs to be extracted, therefore providing enough cues for the software agent to

identify the information. Being a manual process meant that the system was tedious to perform and error prone. Also, the human trainers managed to produce lots of wrappers having high precision but low recall, therefore making the process in general very time consuming.

To overcome this problem, (78) developed Wrapper Induction (WI). In WI a wrapper is defined as being a function from an information source to a tuple and induction is the task of generalizing from labelled examples to a hypotheses. What happens is that "Oracles" are constructed whose task is to recognise instances of particular attributes on a page. There are two types of Oracles, page and label Oracles. The difference is that the former generate example pages specific to particular information resources while the latter is composed from reusable domain independent heuristics. These Oracles are rated according to whether they are perfect (accept all positive instances and reject all negative instances), incomplete (reject all negative instances but reject some positive instances), unsound (accept some positive instances but accept some negative instances) or unreliable (reject some positive instances and accept some negative instances). According to their ratings they are then composed together. The result is a set of labelled pages which is then given as input to the algorithm that builds the wrappers. In this particular implementation, the items of information which need to be extracted are bound using a wrapper made up of a head and tail element which delimits the start and end of the information. Inside these two elements, there are several left and right elements that are used to locate the several instances of the pattern inside the bounds. This wrapper is called a Head Left Right Tail (HLRT) wrapper. The model is a very simple one and basically computes common prefixes and suffixes of sets of strings, which are highly constrained after very few examples.

This can be seen further in (77). In this model, Probably Approximate Correct (PAC) analysis is used in order to identify the number of examples required for a learner to be confident that its hypotheses is good enough. Therefore, whenever this level is reached, the algorithm stops learning. PAC analysis as described in (86; 75; 67) basically states that the error of a hypotheses measures the closeness to the concept being searched for. So, if we have a class of concepts C defined over a set of instances X, our task is to find a learner L using hypotheses space H such that it manages to cover the concepts given a minimum error (ε) and high confidence level (δ). These constraints must be met in polynomial time.

The WI algorithm, although quite robust managed to produce wrappers that cover only 48% of the given data. Although this result is much better than producing all the wrappers by hand, it is still low. Another disadvantage is that to train the algorithm, Oracles must be created, which is not a very intuitive approach especially with naïve users. Also, wrappers are limited to structured data, but most of the web is made up of semi-structured or free text therefore constraining wrappers to limited domains. Finally in (78) it is stated that because the PAC model is too weak, the number of examples needed for efficient learning is high. The consequence of this is that wrapper construction is bound to be an off-line process.

Boosted Wrapper Induction

An enhancement to the WI algorithm is Boosted Wrapper Induction (BWI) as described in (46). Since WI is restricted to highly structured domains, it is unsuitable for most convention IE tasks. BWI uses a technique called boosting whose task is to boost the performance of weak classifiers (in this case the wrappers). First weak rules are learnt based on simple contextual patterns. Since the patterns are applied

on any kind of text, the amount of regularity is typically very low, making the rules high in precision but very low in recall. Therefore to make effective IE several weak rules combined together are used in order to identify a single element. In contrast to WI, instead of having an HLRT wrapper we have a FAH (Fore After Histogram) wrapper. The fore detector identifies the starting boundaries, the after identifies the end boundaries and the histogram is a probability of the field's length. At every stage, the results obtained by these three classifiers are combined and if the combined result surpasses a threshold, it is accepted as being positive. The three classifiers have equal probability of occurring therefore making the result of their combination proportional to a naïve Bayesian estimate with uniform priori (as specified in (86)). From the experiments, handling exact tokens was not generalising enough; therefore the classifiers were further enhanced to handle wildcards. Since both the Fore and After are quite weak detectors, their effectiveness is increased by using Boosting as described in (49). This version of the algorithm is called AdaBoost which is short for Adaptive Boosting. The boosting algorithm invokes the learner using a uniform distribution for all training examples, generates weak hypotheses and then progressively adds the weight of the misclassified examples and decreases the weight of correctly classified examples. Thus the learning tends to focus on the harder examples. The final hypotheses is a weighted majority vote of the weak hypotheses. The algorithm is fast and simple without any need of tuning parameters and it can also be applied to any learning algorithm. Its strength lies in the fact that it manages to identify examples which are hard to categorize while its weakness is revealed when complex weak hypotheses or very weak hypotheses are used. In BWI, Boosting improves enormously the result obtained although the amount of cycles required to reach high performance was very

sensitive to the difficulty of the task at hand. BWI was tested on several domains and in most tasks produced very good results. This algorithm is a living proof of how weak classifiers such as wrappers can be boosted to obtain competitive results.

Another technique worth mentioning is Bagging (Bootstrap Aggregation) as described in (69; 13). This technique has some similarities to Boosting in that they both generate a number of classifiers and uses a voting system, but the similarities stop here. Bagging does not modify the distribution of the training examples but rather keeps on generating new hypotheses based on randomly chosen training examples. In (8) it is estimated that whenever the number of training examples is large, there is a 63% chance that the training set contains unique instances. This means that there is a high probability of having duplicates in the training set and therefore the different classifiers will be trained on a smaller sets resulting in destabilising stable algorithms. The final results from the classifiers are obtained by aggregating in some way the results of the different sub-classifiers normally by taking the average.

Both Bagging and Boosting offer substantial improvements to the learning algorithms, although Boosting generally tends to produce better results. Boosting however tends to be quite sensitive to noise. Both these techniques seem to have a common shortfall, i.e. they both create complex classifiers which may become incomprehensible to users and which do not seem to cover the domain.

SLIPPER

SLIPPER is another algorithm that uses boosting to improve its performance. Its origins can be traced back to the Reduced Error Pruning (REP) method, then Incremental REP (IREP) and eventually RIPPER (Repeated Incremental Pruning to Produce Error Reduction) in (32). The original REP works similar to decision trees

learning systems by first overfitting the training data and then pruning the complex tree. The pruning method chosen at each stage depends on which one reduces the error on the training set. This was later on improved by IREP. The main difference being that the pruning is not performed after all rules are generated but every rule is immediately pruned when it is created. This avoids exploring parts of the search space that are unreachable, therefore making the algorithm more efficient and obviously faster in general. Even though these improvements were noticed, the algorithm still did not manage to get results better than the C4.5 algorithm (86). Because of this, the algorithm was enhanced further and RIPPER was created. It implements three main improvements over its predecessor. First of all, the metric used to guide pruning was altered because it was noticed that the original metric occasionally stopped the algorithm from converging. Second, when IREP was generating rules, it had a threshold heuristic which halted the rule production process if the error exceeded 50%. This made the algorithm sensitive to small problems, because of this, it used to stop generating rules much more earlier than it should have done. To solve this problem, instead of binding the stopping condition with the error rate, it was bound to a minimum description length (mdl) of rules. The threshold value was chosen empirically from experiments. The final improvement was to introduce a re-optimisation at the end of the process (in a similar way to REP) over the whole rule set.

RIPPER finally managed to outperform C4.5 in the majority of tests performed. Not only were the results obtained better, but due to its efficient algorithm, the system in general converged faster than C4.5. A further improvement on RIPPER, which makes use of the same boosting technique that BWI uses, is SLIPPER (Simple

Learner with Iterative Pruning to Produce Error Reduction). The algorithm works by repeatedly boosting a simple, greedy, rule learning algorithm. At every stage, examples are not removed from the training set but their weighing is reduced. A voting mechanism is then used amongst the rules, in order to rate the coverage of the elements in the training set. An interesting fact is that rules have the ability to abstain from casting a vote if the example is not covered by the rule. The learner generates the rules by splitting the training set into two, extracting a single rule using one subset and then prunes the rule using the other subset.

SLIPPER manages to achieve good results. Most of them must be attributed to the benefits obtained from boosting. Also SLIPPER manages to scale well to large noisy data sets even though it is less efficient than RIPPER.

$(LP)^2$

Another algorithm which works on similar lines as WI is $(LP)^2$(Learning Patterns by Language Processing) which is described in (24; 23). This algorithm is an Adaptive IE algorithm; by adaptive it is meant that the algorithm is capable of being ported to new scenarios without the need of an IE expert to configure it. In fact, for the algorithm to work, all that is required is a simple scenario which describes the particular domain. It performs information extraction using linguistic analysis. This information is then used to generalise over the flat sequence of words, but the algorithm goes further and tries to utilise the structure of the document (E.g HTML, XML, etc) depending on the task at hand.

The $(LP)^2$ algorithm induces symbolic rules learned from a tagged corpus and depending on the implementation version it could use either top-down or bottom-up generalisation. It mainly induces two types of rules: tagging and correction rules.

Tagging rules are made up of a sequence of words followed by an action to insert a tag. The word sequence is a window of ω words to the left and right of the tag. For each positive example, the algorithm builds an initial rule. This rule is generalised by dropping some of the constraints (in bottom-up generalisation), introducing some wildcards and introducing some of the results obtained from shallow NLP tools (such as a morphological analyser, Part Of Speech [POS] tagger or user defined dictionaries). The different possible combinations of rules are generated, pruning is used to remove unreliable rules or rules that cover instances covered by existing rules and a search is performed in order to determine which rule is the best. The best rule is determined by checking the error rate against a maximum threshold. Finally, the algorithm keeps the k best generalisations of the initial rule. The instances covered by the k best rules will not be available any more for rule induction and the cycle continues until there are no more instances to cover. The rules obtained at this stage are high in precision but low in recall. $(LP)^2$ tries to increase recall without effecting precision by recovering less good rules and it tries to make them reliable by constraining them. These rules may not be good enough to identify a specific instance of a concept but may be enough to define the boundary of a particular instance. They are called contextual rules. Once rules are learnt, additional rules are induced in order to correct mistakes. These rules do not insert new tags but rather correct any misplacement of tags. Finally, the tags produced are validated; this means that a check is performed to make sure that basic conventions are adhered to (such as every opening tag should have a corresponding closing tag). This algorithm has the benefit of reducing training time, reducing corpus size and limiting, both the data sparseness and overfitting in the training phase. These properties are achieved due to the use of linguistic information. In fact morphology

allows overcoming of data sparseness due to number/gender word realisation while POS tagging allows generalisation over lexical categories. Since more powerful and generic rules are created easily, the algorithm converges much more rapidly. $(LP)^2$ implements a number of innovative and successful ideas which make it compare well with the best IE algorithms available today.

CRYSTAL

CRYSTAL as mentioned in (102) took a different approach than the algorithms already seen. Instead of working towards extracting information, its main task was to produce a conceptual dictionary. A conceptual dictionary is made up of a list of contextual nodes and each node is basically a description of the local syntactic and semantic information in which relevant information is likely to be found. It tries to achieve inductive concept learning which is similar to Inductive Logic Programming (40) by using a specific-to-general data driven approach in order to find the most specific generalisations that cover all positive examples and no negative instances.

The system starts by using a set of tagged documents having the concepts related to a particular semantic class annotated. For all the available instances CRYSTAL starts by inducing a contextual node from the information available in the background. It then goes through all the definitions and compares highly similar examples. Since there may be many rules especially in large corpora, the system makes use of indexes in order to locate quickly similar contextual nodes. The similarity metric used, counts the number of relaxations necessary in order to unify the two structures. The rule which produces the best unification is kept and tested in order to check that it does not cover wrong instances. If the new generalisation is valid, all the rules that are covered by these new rules are deleted from the database and only the new rule is

kept.

CRYSTAL is one of the first systems that tries to achieve the sub-task of information extraction automatically. The system also supplies some tradeoff parameters which can be adjusted in order to tune the precision and recall required. A limitation of this approach is that the examples generated are not generic enough and most probably, will only match the sentence they were learnt on or very similar ones. The major contribution of this system was that it was one of the first systems whereby the user was not required to know anything new apart from information about the domain in which he was an expert.

AUTOSLOG-TS

AutoSlog-TS (94) was a major research contribution toward making information extraction systems more easily portable to new domains. It is the first system that can generate extraction patterns automatically using only raw text as input; other systems require annotated training texts or other forms of specialised training data. Tests showed that its extraction patterns achieved performance comparable to the hand crafted patterns of its predecessor AutoSlog (93).

AutoSlog was a dictionary construction system that created extraction patterns using heuristics. In the system the phrases to be extracted had to be tagged in some way. The rules which were generated were divided into three categories, based on the syntactic class of the noun phrase being examined. Also, after a rule was created, it had to be reviewed by human experts in order to remove any errors introduced by the part of speech tagger.

AutoSlog-TS is basically an extension to AutoSlog. It operates by creating an extraction pattern for every noun phrase in the training set. The patterns are extracted

by using some domain independent heuristics. The extracted pattern is then evaluated by going through the training set a second time and collecting statistics about that pattern. The statistics are used to calculate a probability based on whether the text is relevant and on which rules are fired. This measure is referred to as a pattern relevance rate. The patterns are then automatically ranked using a formula that promotes high relevance or high frequency.

In this system, the extraction patterns managed to produce significantly fewer spurious extractions and better text categorisation results than those generated from the hand-crafted patterns on one of the categories. Using AutoSlog-TS, the extraction-based text categorisation algorithms can be trained for new categories much more easily than before. This is the first algorithm aimed at building semantic lexicons from raw text without using any additional domain knowledge. A user only needs to supply a representative text corpus and a small set of seed words for each target category. The experiments showed that a core semantic lexicon can be built for each category with only 10 to 15 minutes of human interaction. The experiments showed that extraction-based categorisation can perform well on fine-grained categories and can be used effectively to classify texts with respect to subclasses within the context of a single domain.

Inductive Logic Programming

A different approach adopted in Information Extraction makes use of a technique known as Inductive Logic Programming (ILP). This technique as defined in (86; 35) and has its foundations as a cross between logic programming and machine learning. It concerns with the development of methods and tools for the automation of reasoning and makes use of a set of algorithms in domains which have a rich relational structure.

In these domains instance attributes are not isolated, but are related to each other logically. Relational learners are basically rule learners which derive features on-the-fly and which can in theory logically derive new features from existing ones.

One of the first such systems and whose basis lies at the heart of many modern ILP systems is FOIL as described in (92). FOIL learns how to construct a logic program that defines relations in terms of it and other background relations. Since it works with first order logic, it uses tuples that belong (or not) to the target relation. It starts with a general clause and progressively adds literals until all negative examples are excluded. The algorithm makes use of a divide and conquer approach, whereby the covered examples in the target space are separated and the algorithm directs its attention to the remaining concepts. At every stage, one or several clauses are created made up of literals. The literals can be either gainful or determinate. The former literals are evaluated using an information gain heuristic that helps to reduce the search space by eliminating unreliable information. The latter on the other hand does not help to reduce the search but rather, adds new variables required for the final clause. During the search, a greedy approach is taken; but this has the potential to explode the size of the search space. In order to avoid this, checkpoints are established along the paths being searched, and if the current path does not lead to an improved condition, the search is halted and does not continue from the last node in the search tree but rather from the last checkpoint, therefore reducing the amount of searches required in less valuable paths.

There are a number of systems related to FOIL. One such system worth mentioning is mFoil (40). This particular implementation improves over FOIL by implementing several techniques to handle noisy data. It also replaces the existing greedy search

found in FOIL with a more efficient beam search, therefore increasing the chances of finding a good clause. This system and may others similar ones form the backbone of modern ILP approaches.

RAPIER

The Robust Automated Production of Information Extraction Rules (RAPIER) is one of the modern ILP systems used for IE. It was built around 1997 (19) and it incorporate techniques from several ILP methods. RAPIER was designed mainly to cater for semi-structured text and it expects as input a pair of documents at a time. One document contains the data while the other contains filled templates. The filled templates hold a list of information to be extracted from the document. The learner's task was then to find in the data the elements specified by the filled templates and learn them using its algorithm. The user was not involved in the process apart from supplying the pair of documents for training.

The algorithm uses a bottom-up approach starting from a specific rule that matches a target concept in the training set. The rule is made up of an unbound pattern made up of constraints on words and part-of-speech tags surrounding the filler. Once the rules are collected, a cycle begins where pairs of rules are randomly chosen and a beam search is performed to identify the best least general generalisation that covers both of them. At this stage, constraints are systematically added and the rule is evaluated. This iteration continues until no progress is recorded.

The rules of RAPIER are based on both content and delimiting information. The patterns exploit syntactic and semantic information in order to have better generalisations. For syntactic information, a tagger is used to obtain the Part of Speech found in the document while for semantic information, a lexicon of semantic classes

is used. The pattern is nothing more than a sequence of items which match either one word or a list of words. Every rule is indexed by a template and slot name, and consists of a pre-filler, filler and post-filler. The current approach handles only single slots although Soderland in (101) proposes possible alternative strategies in order to extend the current paradigm to perform multi-slot extractions. The idea is to divide the text into more than just three fields.

SRV

Sequence Rules with Validation (SRV) as described in (45) is a general-purpose top-down relational learner for IE. The system works by taking as input a document containing markup information around the data to extract. It makes no assumption about the information available for use and the document is treated as a token-oriented feature set. The features range from token length, type, orthography, POS, lexical meanings and other relational structures. These can be divided into two classes, simple and relational. A simple feature is a function that matches one token to some discrete value. A relational feature provides a mapping between a token and another token such as adjacency. The information is extracted from the token level because multi-term text fragments are difficult to describe in terms of simple features. The output of this process is a set of rules which are used to extract other similar concepts.

This algorithm looks at IE from a classification point of view and considers the document as being made up of several mini-documents where every candidate instance is presented to a classifier and rated according to its relevance which is a confidence level that shows how probable the phrase is part of the target concept. The search is performed in a similar way to how FOIL does it in [QUI95]. It basically goes

through a test space made up of all the textual fragments from the test document. The extraction is performed by examining the fragments of appropriate size in order to match them with the available rules. The learning process is once again similar to FOIL, i.e. the data space is divided into two pools containing negative and positive examples. The positive examples are those elements tagged beforehand while the negative examples are the remaining elements. Induction is performed and only stops whenever a rule does not manage to cover more elements than the previous rule. At this stage, the rule is considered to be good and all the examples covered by the rule are removed from the training set and the process goes on until all examples are covered.

This algorithm has quite a number of potentials. First of all no prior syntactic analysis is required for it to work. It also makes use of a very expressive representation using orthographic features and linguistic information when available. A limitation of this algorithm is that it was designed to extract single-slots.

WHISK

The WHISK system presented in (101) is a covering algorithm which performs top-down induction of rules. It is based upon ILP principles but with a difference from other systems mentioned so far, in that it is capable of handling any kind of text. It can extract information from structured pages, from semi-structured texts like web pages and free text. When it comes to free text, WHISK can also make use of a syntactic analyser and semantic tagger to increase its performance.

The WHISK learner works in a supervised mode with phases that alternate between tagging and training. The learner starts from a set of untagged instances and an empty training set. In each iteration, the system selects and presents to the user a set

of instances for annotation. The definition of an instance is variable in this context. It depends mainly on the kind of text being analysed. If the text is semi/structured having some sort of tags, heuristics (mainly based on HTML tags) are used in order to divide the text according to the position of specific tags, while if the program is handling free text a sentence splitter is used to divide the document into sentences.

The user's task is then simply to mark the region of text where the information for extraction is situated. When the user is ready, the batch is sent to the learner in order to induce a set of rules from the new expanded training set. Since WHISK is a top-down learning algorithm, it starts the induction by finding the most generic rule that covers a seed concept. Terms are then gradually added until the error is reduced to zero or until a pre-pruning criteria is satisfied. The iteration finishes when the generated rules manage to cover all the positive examples in the training set. Some post-pruning is also performed in order to remove some of the rules that overfit the data.

The main limitation of WHISK is that its feature set is not very expressive and therefore its generalisations are less effective than other learners. The main advantage which it clearly has over the others is that it can manage to extract multi-slot information, but this comes with the added costs of requiring more examples than other learners.

SNoW-IE

SNoW-IE is the last ILP based system this subsection will look at. It is describe in (97) and is based on a new paradigm for relational learning which promises to be more flexible and allows the use of any propositional algorithms including the probabilistic ones.

The problems with such systems has always been to learn definitions of relations or concepts of interests in terms of given relations. Although there were systems capable of doing this, they were not scalable enough to large domains and consequently might not generalise well. Research in ILP suggests that, unless the rule representation is severely restricted, the learning problem is intractable. The main problem with such systems is that most of them have problem specific functions therefore restricting such systems to specific domains. The SNoW-IE system makes use of efficient general purpose propositional algorithms. At the heart of the system, there is a knowledge representation language that allows an efficient representation and evaluation or rich relational structures. The language is a subset of First Order Logic and works as a collection of graphs defined over elements in the domain. Restrictions are imposed by limiting the expressibility of the language to a collection of formulae that can be evaluated efficiently. Objects are also given types in order to classify objects in the language according to their properties. The language to do this creates Relation Generation Functions (RGF) as described in (33). These functions generate more expressive formulae that model the structures in a domain. In this proposed calculus a formula is only formed when all its parameters are available and therefore needs not be tested in order to see if it can fire.

The main difference with respect to previous ILP approaches is that the search space is structured in a different way, instead of generating the features as part of a search procedure, they are obtained in a data driven way using the RGFs. Knowledge is incorporated in the system by using sensors. These programs have the ability to treat information from input, external sources or previous information in a uniform way. The data at the back of the system is expressible enough to be used by any

system therefore making the paradigm not bound to any specific learning algorithm. To test the paradigm SNoW (Sparse Network of Winnows) (96) was used. The learning architecture is a multi-class classifier that is specifically tailored for large scale learning tasks and for domains in which the potential number of features taking part in decisions is very large, but may be unknown a priori. It learns a sparse network of linear functions in which the targets concepts (class labels) are represented as linear functions over a common feature space. The system offers a feature efficient learning algorithm, in that it scales linearly with the number of relevant features, and linearly with the number of features active in the domain.

The improved system called SNoW-IE first starts by filtering the negative examples. This part can be seen as a classifier designed to obtain high recall. The examples are removed if they do not contain a feature which should be active in a positive example or if a confidence level is below some threshold. The second part is aimed towards increasing precision and is called the classification stage. For this stage, the candidate fragments are enhanced using a new collection of RGFs, then a classifier for each concept is trained. Training results show that the system outperformed most rule based systems. The system works in a similar way to existing ILP techniques but instead of representing the rules as a conjunction, they are represented as a linear function.

Maximum Entropy Classifier

An application of Maximum Entropy Classifier to IE is described in (22). This approach is similar to SRV in that it too looks at the problem as being a classification problem. The Maximum Entropy (ME) principle is very old and simple. Basically it models all that is known and assumes nothing about that what is unknown. In

other words, given a collection of facts, choose a model which is consistent with all the facts, but otherwise as uniform as possible (9). Also, the ME principle takes into account features that are not independent therefore making it suitable for multi-slot extraction.

Every candidate for a slot filler can be one of five types i.e. a begin slot, a continue slot, an end slot, a unique slot, and a not-a-slot. To perform extraction, several group of features were defined. These groups include unigrams, bigrams, zones (delimited by tags found in the document), capital letter words, headings, time, names (according to a gazetteer) and new words (not found in a dictionary). The features are associated to the words in the document and a probability is associated to every feature using the ME classifier. The Viterbi algorithm is then used to select the sequence of word classes with the highest probability. This approach is typically used to perform single-slot extraction.

For multi-slot extraction, some enhancements were added. First of all, the texts were passed to a text-filtering module whose task was to remove irrelevant documents. The documents were then semantically tagged so that information such as people's name, organisations etc are found and tagged in the document. This phase was called candidate selection. The third step is to produce relation classification. Basically, it is the task of finding relations between entities. Finally, a template building procedure is used. It takes a graph of relations between entities as input and fills a template using the relations in the graph.

This approach is capable of handling both single and multi slots for both semi-structured and free text. The results obtained were superior in most cases to any algorithm available. The limitations with this approach has mainly to do with how

the features were selected. Some features defined were dependent of the domain being analysed, yet this does not diminish in any way the potential of this algorithm because domain independent features can still be produced even though the success of the algorithm will probably be less.

HMM

A Hidden Markov Model (HMM) is not the name of an IE algorithm. It is a mathematical model used in various areas of computer science. In (48) a HMM was used to locate textual sub-segments in a document that answer a particular information need. To build these models, a statistical technique called shrinkage was used to improve the estimation of the parameters of the model.

In this implementation HMMs are used as a probabilistic generative model of the entire document. The documents were not fragmented due to the fact that it could cause a huge space of fragments. To find the most probable sequence of elements, the Viterbi algorithm was used to learn the model transitions and emission probabilities from the training data. Separate HMMs were built to extract different fields with each HMM having two states:

- Target state containing the tokens to extract

- Background state containing the remaining elements

The HMM is not fully connected (In a fully-connected graph, each cell is just one step away from every other cell), the reason being to capture textual context therefore improving efficiency. The state transition topologies are set manually and not learnt. Two topological parameters which were tested are the window size (Number of prefix and suffix states) and the target path count (Number of parallel, different length

and target paths). In the model only target states can emit target tokens and vice-versa for non-target states and only a single unambiguous path is possible through any topology. In order to improve the parameter estimation Laplace smoothing and Absolute discounting were tested. Both techniques make use of only the training data available.

Shrinkage is a statistical techniques used in speech recognition systems. It makes use of weighted averages and learns using the Expectation Maximisation algorithm. Basically, it shrinks parameter estimates from data sparse states of complex models towards estimates in related data-rich states of simpler models. In some conditions, this algorithm was shown to produce optimal results.

To perform IE, the system was defined in terms of a hierarchy, i.e. subsets of states were identified with similar distribution (Eg: all prefixes are expected to have the same distribution). This was done to represent a similarity between parameter estimates. At the top of every hierarchy is the uniform distribution (which gives all words equal probability). States were further divided into four classes, these are Non-target, Target, Prefix and Suffix. There were also four shrinkage configurations:

- None (use only absolute discounting)

- Uniform distribution

- Global (shrinkage is performed towards a common parent and eventually the uniform distribution)

- Hierarchical (the target class are shrunk in a similar way to the Global configuration, while the other classes are shrunk towards a separate parent. Prefix and

Suffix are shrunk towards a "context" grandparent. In turn every state is eventually shrunk towards a common ancestor and eventually uniform distribution)

EM Technique is finally used to re-estimate the parameters and the cycle continues.

Experiments using this system showed that as the size of the window increases precision decreases. The more complex model fractures the training data making it difficult to obtain parameter estimates. Shrinkage improved the model over large window size. It managed to reduce the error of parameter estimation by 40% and allowed the learner to learn more robust HMM emission probabilities using limited training data. It was also shown that the most consistent patterns were the closest to the target concept because the local weight of states declined with increasing distance and when related to field difficulty.

ESSENCE

ESSENCE (20; 21) is one of a different breed of IE systems that promise to build IE patterns automatically based on an un-tagged corpus of relevant documents. This task is achieved by using a generalisation algorithm which delays as much as possible the involvement of the user in the system. This delay occurs because the system will gather as many patterns as possible and the user is then asked to go through the potential patterns and not through the whole corpus.

The ESSENCE methodology makes use of a general purpose lexicon. In the current implementation WordNet (84) is used to provide lexical, syntactical and semantical information. Since WordNet is not an exhaustive list of world concepts, additional gazetteers and domain specific word lists are used to complement WordNet. The first step in the methodology is the **Task Definition** module. The user is asked to enter a number of keywords that commonly appear next to the target concept. For each

keyword, the user is also asked to select the synset number as defined in WordNet. With this information, the system calls the `Selection of Relevant Texts` module whose task is to select from the corpus a set of text based on the user's previous input. Noun phrases, verb phrases and prepositional phrases are then identified because they will form the basis of the generated patterns.

The sentences obtained will then be filtered according to a relevance judgement based on the number of keywords found in them. Out of the relevant sentences a window is selected which will be used for analysis. The nouns and verbs are then semantically tagged using WordNet. At this stage, the patterns are learnt by the ESSENCE Learning Algorithm (ELA) and the resulting patterns are filtered in order to remove very specific patterns. The ELA works similarly to other covering algorithms. The main difference is that the granularity of elements is much bigger, working at the noun, verb and preposition group level. Also the generalisation is performed by relaxing the semantic tags associated to groups of patterns.

Although the results obtained by this system were quite respectable, the true achievements are twofold. First of all the system manages to obtain such results using a higher granularity than the one used by other similar systems. Secondly, it does so without requiring lots of human intervention. Basically annotation is not required, only a number of seed words are needed to start the IE process.

LIEP

The LIEP (Learning Information Extraction Patterns) system (71) is an IE system which aims primarily towards the effective acquisition of patterns. It learns the extraction patterns using an On Demand Information Extractor (ODIE).

The ODIE first tokenises the text given as input. It then goes through all the

sentences, a sentence at a time and checks whether some of the keywords (which indicate the possibility of an event of interest) appear in the current sentence. Those sentences which have no keywords are discarded while the others are kept for further analysis. The retained sentences are tagged with a part of speech tagger and a pattern matcher is run over the sentences in order to identify entities and other information (like prepositions, noun groups and verb groups). The system then tries to identify events by using IE patterns that match syntactic constituents and other properties. At no stage does the system attempts to perform a full parse but simple syntactic relations are checked by using simple rules to verify the validity of sub-constructs found in a sentence. This technique is called on-demand parsing. The advantage of such an approach is that the burdens associated with a full parse are not required but this obviously means that the text is not adequately constrained as in a full parse and this may lead to over-generalising some of the rules.

The actual patterns are learnt by going through the different elements found in the system and trying to identify relations between them. In the case that several paths exist between the elements, all the paths are generated. The patterns are then tested and the one with the highest F-measure is selected. The selected patterns although high in precession would have a low recall since they are too specific to the text in the given document. Because of these, the patterns are generalised further.

LIEP in general compared fairly well with IE tools developed during the same time. Its generalisation was not optimised and its main source of input was only from positive examples. Negative examples were only used for testing purposes and no additional use was made out of them.

CICERO

The CICERO system (66) is an IE engine based upon the effective utilisation of WordNet (84). The idea is to mine WordNet in order to discover concepts and lexico-semantic relations relevant to the current domain. To check the validity of these relations, tests are performed using the domain dependent documents that are available.

The methodology implemented in CICERO first involves creating a semantic space that model the specific domain via WordNet. The semantic space is in the form of a triplet made up of concepts, connections and contexts. A concept list is made up of words or group of words all referring to the same concept. Connections can be of three types, i.e. thematic, subsumption and contextual. Thematic connections refer to elements related by lexico-semantic or morphological relations. Subsumption connections refer mainly to Is-A relations found in WordNet. Contextual connections define possible relations between elements in the same context. The final element in the triplet is a context element. The context element models the different ways and restrictions in which different words and objects can be arranged, in order to express a concept. This is done due to the nature of natural language whereby a concept can be expressed in a multitude of different ways.

This space restricts enormously the searches required in WordNet when modelling the domain. The documents being analysed are then scanned to identify and link the domain concepts which are defined in the semantic space. The patterns obtained are finally classified against the WordNet hierarchy and only the most general linguistic domain patterns are retained.

For CICERO to work, several algorithms and a number of heuristics were devised

	Speaker	Location	Start Time	End Time	All Slots
$(LP)^2$	**77.6**	75.0	99.0	95.5	**86.0**
BWI	67.7	76.7	**99.6**	93.9	**83.9**
HMM	76.6	78.6	98.5	62.1	**82.0**
SRV	56.3	72.3	98.5	77.9	**77.1**
Rapier	53.0	72.7	93.4	96.2	**77.3**
Whisk	18.3	66.4	92.6	86.0	**64.9**
SNoW	73.8	75.2	**99.6**	**96.3**	85.3
Max Entropy	72.6	**82.6**	**99.6**	94.2	**86.9**

Table 3.1: F-measure on CMU seminar announcements.

in order to make effective use of WordNet. WordNet can be seen as the strength and weakness of this system. Whilst it provides that syntactic and semantic knowledge which is so difficult to obtain, its topology makes it difficult for users to retrieve such information.

Discussion

It is difficult to compare the different algorithms together without examining how they perform when confronted with a common task. Luckily, experiments on the CMU seminar announcements collection[1] were performed by most of the algorithms mentioned in Subsection 3.2.2 and they were reported in (24) and (22). The CMU collection consists of 485 seminar announcements and the task involves identifying the *speaker*, *location* of the seminar, the *start time* and *end time*. The documents have 485 instances of *start time*, 464 *locations*, 409 *speakers* and 228 *end time*. The results of the different algorithms can be seen in Table 3.1.

The outcome from these experiments is very interesting. We will mainly compare the algorithms which produced the most promising results. First of all it can be

---

[1]Downloadable from http://www-2.cs.cmu.edu/ dayne/SeminarAnnouncements/

noted that $(LP)^2$ and BWI produce comparable results even though in the final score $(LP)^2$ outperforms BWI. The reason for this is that both algorithms are based upon the same concept of creating wrappers. $(LP)^2$ results are better in two particular instances, *speaker* and *end time*. The reason being that a *speaker* can have a very variable context. Therefore, the linguistic features used by $(LP)^2$ are capable of modelling this variable context in a better way than BWI. When it comes to the *end time* concept, from the distribution of the concepts in the documents, it can be noticed that it is the concept with the least instances. In $(LP)^2$ it was noted that a positive side effect which linguistic features add to the learning of the algorithm is that they tend to reduce spurious data and this is the reason why the algorithm manages to get a higher score for *end time*. BWI would require more examples in order to improve its score. The results of BWI are better when it comes to *start time* since the collection contains many examples of that concept and therefore the algorithm is capable of boosting the wrappers in order to model the data more effectively.

The two other algorithms that performed well in the tests were SNoW and Maximum Entropy (ME). Both of them are based on different genre of algorithms. SNoW is based on a new form of ILP which is much more generic than common approaches. This form of relational learning produced good results when it comes to *start time* and *end time*. These two concepts are very regular but an interesting fact is that SNoW manages to get the best result in *end time*. This means that this form of relational learning is capable of learning regular patterns using very few examples and therefore overcoming the problem of spurious data. The ME method too performs well on the *start time* but the most surprising result is that it manages to get the best result for the *location* concept. It manages to outperform the next best algorithm by about

4%. This can be attributed to how ME works. Basically it models the domain using the positive examples only and makes no assumptions on the remaining parts of the document. This domain model is more representative than any model produced by the other algorithms and in fact, the best result for this concept after ME is produced by the HMM algorithm which is based on a similar idea of domain modelling.

3.2.3 Deeper Approaches
LaSIE

The LaSIE (Large Scale Information Extraction) system is one of the few systems that make use of deeper approaches to perform IE. The initial version of the system called LaSIE-I (53) is an integrated system that performs lexical, syntactical and semantical analysis to build a single, rich representation of the text which is then used to perform several IE tasks.

The lexical analysis is very similar to the analysis performed in shallower approaches. The text file given as input, is passed through a tokenizer, sentence-splitter, part-of-speech tagging, morphological analyser and a pattern matcher. The pattern matching works by checking the input against a number of lexicons that contain lists of organisations, company designators, people titles, human names, currency units, locations and time expressions. If an element in the document matches one of the elements in lists, a category is assigned to that element equivalent to the name of the list in which it was found. Apart from these lists there is also a hand crafted list of trigger words used to give an indication of the possible class of the surrounding tokens.

The syntactical analysis takes as input the data obtained from the previous stage and parses it using a simple Prolog parser. The parsing occurs in two passes. In the

first pass a Named Entity Grammar is used in order to locate domain specific named entities. In the second parse Sentence Grammar Rules (extracted from the Penn TreeBank-II (83)) are used to extract a semantic representation from the sentences.

The semantic analysis first starts by passing the output from previous stages to a discourse interpretation module that builds an ontology using the XI knowledge representation language (51). The ontology also has associated with its nodes attributes therefore making the final result a *world model*. Apart from attributes, the ontology may also have inference rules associated with its nodes. When the ontology is being constructed, a co-reference resolution module tries to resolves any references in the text. This module has a number of heuristics which try to resolve the several instances found in the text.

The results are gathered from the system by simply going through the *world model* and gathering the required information. Since the MUC-6 results of LaSIE-I were quite promising, the team behind it created an improved version called LaSIE-II (72) and submitted it for MUC-7. The main difference between the earlier version is that all the modules are organised and executed as a pipeline inside the GATE architecture[2]. Other minor differences include

- enlarging the lexicons available with new data.

- sentences are represented in Quasi-Logical Form (QLF) (52; 5). This meaning that they are expressed as conjunctions of first order logical terms.

- adding presuppositions to the data such as additional inferences, additional hypotheses, word sense disambiguation, person role classification and partial parse extension.

---

[2]http://gate.ac.uk/

- adding object co-reference with extra features like long distance co-reference, copular constructions, cataphora, coordinated NPs, header co-reference and generic nouns.

- adding consequence rules used mainly to fill slots in the results.

- performing event co-reference in order to merge events with additional information found elsewhere in the text.

LaSIE-II does not really differ much from its predecessor in terms of the approach adopted. Even so, the small refinements in the system managed to produce improved results on a much harder test.

LOLITA

The LOLITA (Large-scale, Object-based, Linguistic Interactor, Translator and Analyser) system (88; 54) is in reality a general purpose NLP system although for the MUC competitions it was used as an IE system. It is mainly composed of two parts, the first part converts the text to a logical representation of its meaning while the second part generates the results.

At the heart of the whole system lies a semantic network. All the analysis tasks are organised in a pipeline and the several output results obtained from the different stages are used to update the information in the semantic network. The network is used to hold different kind of information such as conceptual hierarchies, lexical information, prototypical events (that define restrictions on objects) and other general events. The rest of the information found in the network is obtained directly from WordNet (84).

The main process first passes an SGML module to a Text Pre-processing module. The module locates structural information in the document such as reported speech, paragraphs, sentences and words. The output is then passed over to a Morphology module whose first task is to normalise the input such as expanding abbreviations, removing hyphens, expanding monetary and numeric contractions. Only the root of a word is retained and to avoid that information such as number, case, person, etc would be lost, this information is stored in a feature structure. After this module, the information is passed to a parser made up of the following five submodules:

Pre-parser used to identify and provide structure for monetary expressions.

Parser used to parse whole sentences. The result of this is a parse forest (made up of all possible parses).

Decoding of the parse forest utilising the previously generated forest and making use of a number of heuristics to select the best trees.

Best parse tree selection takes as input the decoded forest selects the lowest costing tree.

Normalisation of the syntax tree done to reduce the number of different cases possible. This normalisation does not alter in any way the semantics of the sentence.

Once the syntactic analysis is completed the system performs semantic and pragmatic analysis. The semantic analysis is performed by going through the trees and building up meaning based upon the tree's subtrees. It makes use of the composition principle that the meaning of a sentence is made up from the meanings of the words contained inside it. The pragmatic stage is used to disambiguate and type check the

data. Once an event is disambiguated, the system attempts to create connections between it and the previous events. While the document is being processed, in the case that the system identifies an anaphoric expression, it looks for possible referents. If it manages to find such references it unifies with them the information it possess. In the case that no references were found, a new instance is created in a reference database called a Context buffer.

The results obtained by the LOLITA system were not great and this was due to a number of limitations of the system. The system requires a successful parse in order to analyse the text. Such a parse was not always possible even though error recovery routines were implemented. Another limitation is that the data required by the system is much larger than the amount provided for testing purposes. Because of this, the results obtained were much lower than the ones expected by the LOLITA team.

Machine Learning Based Text Understanding System

The Machine Learning Based Text Understanding System (103) is a system designed for very deep textual understanding. The domain for which it was developed is the medical domain in which it is vital to have full understanding since a patient's life may be at stake.

The system accepts as input a textual description. This is passed to a structural analyser whose task is to make use of a maximum entropy classifier to determine the sentence boundaries. The sentences are then passed to a lexical analyser. It makes use of a handcrafted domain specific lexicon to assign syntactic and semantic features to specific words. A syntactic parser is then used to create a dependency graph linking each word to the word it modifies. The probability of every arc is obtained from a

statistical parser trained upon hand-tagged training sentences. After this stage, a semantic interpreter (trained upon handcrafted rules) is used to go through the data and search for semantic relations. These relations are then grouped into a frame. In this context, a frame is basically a factual entity with a list of properties associated to it. The final step is a co-reference resolution module that tracks references amongst frames in sentences. Co-reference is performed using hand-coded probabilistic rules and lexical cues.

Although such systems in reality are far from full text understanding, the methodology developed here is a step forward. The system has a serious limitation when it comes to porting to new domains. Some domain specific lexicons must be created and a system developer must create a precise definition of the target concepts for the domain.

PENSÉE

With difference to the systems seen so far, *PENSÉE* (50) is a commercial system developed by Oki[3]. The original modules used in this system were constructed for Machine Translation (MT) purposes. In fact, one of the reasons Oki designed this IE system was to find out how effective the core modules are when used in other applications. The system is made up mainly of two modules. A surface pattern recognition module and a structural pattern recognition module.

The surface recognition module detects named entities and co-references without any lexical or syntactical analysis. This module searches for capitalised areas and merges groups of words whenever there is a high probability of having a compound word (based on some handcrafted heuristics). Types of named entities are recognised

---

[3]http://www.oki.co.jp/

using other rules by making use of cue words found before or after words of specific types. The remaining capitalised words which are still ambiguous are checked against a handcrafted dictionary.

The structural pattern recognition module traces parse trees and uses them to find named entities, co-references, etc. All these are expressed using the Grammar Description Language (GDL). The GDL system allows the user to describe extraction rules used during the IE process. These rules are then translated and interpreted. The rules are basically divided into two parts, a matching part and an action part. The matching part describes a condition to match a part of a tree representing the document and the action part describes what changes need to be done to the tree. Also, to help the user create such rules a powerful debugging system is supplied with the tool.

The system in general did not obtain good results in the evaluation even though to be fair, the developers emphasised that this system was developed in a very short time. Most of the rules used to identify the several concepts were handcrafted and the amount of automation found in the system is very scarce. Even though it has all these faults, in itself it is a simple and interesting system that shows how pattern matching can be effectively used in order to extract more information without the need to make use of deeper approaches in some cases.

3.3 Motivation for using II

The vast amount of data available on the Internet unleashes an unimaginable number of ways in which the information can be combined and utilised in new applications and tools. Even though this may sound simple, when an automated agent tries to exploit the available data, it is faced with many obstacles. Imagine you are driving in a city without well defined roads, where the roads and buildings are changing as you move around. The road signs, even though they mean the same things are represented differently in different districts of the city. It sounds like an extract from the 1998 science fiction film **Dark City** but this is the current situations automated agents must face when they surf on the internet. One of the main reasons is the lack of structured representation available in the information found in web pages. This irregularity makes the data too complicated to navigate. The problem gets more complex when we consider that part of the data which is online, is not static, many web pages are dynamically generated on the fly from scripts or databases which change frequently. What is referred to as static data changes as well every so often. Links which existed a second ago go suddenly dead and others which were not there appear as if from nowhere. Another complication is that there is no uniform vocabulary to describe the same concepts amongst different domains or even the same domain.

Apart from this, the II technologies available are not mature enough since they normally require the construction of specialised applications in order to extract information from the different sources. This obviously makes the whole process

time-consuming since it is a very repetitive task performed by a human. Also, the side effect of having a human do the job is that a human is also error prone and costly.

difficult to maintain since the structure of a page can change from time to time. Every time this happens, these programs are broken since they were trained upon a particular structure which does not exist any more.

Therefore, the following sections will focus on the basic foundations and techniques in Information Integration as it applies to the Web.

3.4 Previous Work in II

This section will give a brief overview of the different types of Information Integration systems available. It will give a flavour of what we believe are the most relevant systems to our work.

3.4.1 Ariadne

Ariadne(6)(76) is an Information Integration system that makes it fast and cheap to build automated agents used to access, query and integrate existing web sources. It uses several approaches borrowed from knowledge representation, machine learning and automated planning. The system also provides tools for maintaining these agents by making it easy to customise them as new resources are available. Ariadne is made up of the following components:

Application Domain Model is an Ontology of the application domain. It is used as a basis to integrate all the information gathered in one single view, by providing a unique and standardised terminology for the concepts of the domain.

Wrappers are important because Ariadne needs to know what information can be extracted from the web page. This information is normally stored in semi-structured web pages. In order to do this the system uses both a syntactic

and a semantic model of the information found on the page. This information is extracted by making use of a simple annotation interface, whereby the user is asked to mark examples of the concepts to extract in the source document. Once the annotations are inserted, a learning algorithm is used to induce the grammar rules for that particular page.

Information Sources descriptions are used to model how the system should navigate through the internet in order to extract the information found distributed across different web pages. The system makes use of the same wrapping techniques described before but the information is not just labelled as atomic data and left there. Associations found in the ontology are used to build a plan of how one item of information can lead to another item in a different page.

Query planner handles the different queries possible together with their decomposition into sub queries. It is all based upon the Application Domain Model since it contains a reference to all the data available. The first step before the actual plan is constructed, is to build the actual integration axioms. This is done off-line so that the costs of creating them during query execution is eliminated. Ariadne manages to extract the different rules by analysing the domain model together with the source descriptions and by calculating the different ways in which these two are related. Once this process is finished, it is time to get a query from the user and process it. The query is first of all parsed, simplified and rewritten so that it matches with the different concepts found in the domain model. It is then passed to an optimising module in order to remove any inefficient queries and finally it is executed over the domain model.

Ariadne works fine for most domains but it has a number of major limitations. Currently the system cannot create wrappers for unrestricted natural language texts. It is bound to structured or semi-structured texts therefore restricting the information which it can obtain. Apart from that, the wrapper construction techniques they use requires a lot of examples, making the whole process quite tedious for the user setting up the system.

3.4.2 KIND

KIND(81)(62) stands for Knowledge-based Integration of Neuroscience Data and it focuses mainly on integrating information from large databases. It is described by the authors as being a Mediator system whereby it acts as a common interface between the several data sources available. This has the benefit of providing a unified view for the several databases, therefore allowing an easier and more effective interface for the users. At the heart of the system, there lies a semantic model of the information sources used to tie together the data obtained from the different sources. The mapping between the different data (called semantic integration) cannot be done automatically and a mediation engineer must construct the integration views for the system to work. The problems faced in this system are different from the traditional ones. Normally systems work in domains with similar concepts and attributes, but in the Neuroscience domain (which is the main domain on which the system is based), this is not necessary the case and different sections of the field have very few concepts and attributes in common. The integration is performed using a logic language therefore increasing the complexity of the system since only a person with experience in logic languages can create these integration rules. Once again the complexity of this system increases since the end users queries are expressed using an XML based query language. The

system manages to perform its task but the setting up of the different sources and its use is not straight forward and requires a certain degree of expertise.

3.4.3 PROM

PROM(37) stands for Profiler Based Object Matching and is a system that actually takes care of matching concepts and attributes in order to find out how the data should be integrated together. The problem is called object matching and seeks to find out if two given objects refer to the same entity in the real world. There are two main novelties in the system. First of all, it makes use of profilers. These profiles contain within them a sort of ontology which is very specific to a particular domain with rules and relationships. The data which is extracted is validated before it is integrated, thus making it possible for the system to reduce the errors. This is important in cases where the data does not make sense in the real world, e.g. imagine finding some information about Mr X which says that his age is 10 and he holds a driving license. For an automated system, there is nothing wrong with that, but for humans who know how things work out in the real word, the above statement does not make sense since it is obvious that no 10 year old can hold a driving license. In these cases profilers come to the rescue by providing real world rules in order to validate the date being integrated. The profiles can be added, removed and reused therefore making the whole architecture very flexible and easily extensible. Secondly, it does not only seek full attribute matching. In some cases, the attributes in a site do not exist in a similar site. Therefore a strength of the system is that it handles partial matches of attributes. This is achieved by using once again the profilers. The system has two sets of profilers, the profilers in the first set are called hard profilers and the others are called soft profilers. The difference is the following, hard profilers contain heuristics

that must evaluate to true whenever they are confronted with matching of attributes or concepts, otherwise the match is rejected. Soft profiles on the other hand express a confidence score and the system works by using a sort of voting mechanism. When all the soft profilers express their view about the matching process, if the confidence level is greater than a predefined threshold, then the match is accepted, otherwise it is rejected. The approach described works fine when the profilers are defined by domain experts, but normally it is infeasible to make use of domain experts to build these profilers. This technology is promising since the results they obtained were quite good, but research into automated methods to create these profilers must still be conducted before these profilers become usable.

3.4.4 Name-Matching

An important task in II is the filtering out of duplicate values from the main database where all the information is stored. Due to the redundancy of the web the same data can be found in different sites around the internet, therefore it is justifiable for the different II and IE strategies to return the same data. In (31) several approaches are explored in order to find out which technique manages to filter out the data while keeping the error rate to the minimum. Finally they propose a technique which is ideal for domains where the data is not structured and little information regarding the particular domain is known a priori.

These functions normally calculate a distance between two strings based upon the number of insertions, deletions and substitutions required to be performed on the first string in order to become like the second string. There have been several distance functions, one of the most popular being the Levenstein distance. These functions normally use characters but there are others such as the Jaccard similarity

that makes use of tokens. A string is considered as being a bag of words in this case called tokens. These techniques work fine for the majority of cases but obviously they are not perfect since there are no semantics or real world heuristics to verify the match.

3.4.5 StatMiner

StatMiner(91)(90) is a system based around a set of connected techniques that estimate the coverage and overlap of statistics while keeping the amount of statistics needed under control. To do so, the system computes statistics for classes of queries rather than for individual ones based upon the attributes being selected to execute the query. In order to do so, the system makes use of hierarchies which classify the attributes and queries. These hierarchies are either designed by a knowledge engineers or induced by machine learning algorithms. Since the classes of queries and attributes can be incredibly huge, the system prunes away the queries which are rarely used therefore reducing the general complexity of the system.

The system was tested on a Computer Science scenario related application called BibFinder. It integrates several sources that provide information regarding the Computer Science domain such as CSB[1], DBLP[2], Network Bibliography[3], ACM Digital Library[4], ScienceDirect[5] and CiteSeer[6]. BibFinder acts as a mediator between all these sources in order to provide a unified view. Each source does not provide the same information, the information extracted from different sources can be missing,

---

[1]http://liinwww.ira.uka.de/bibliography/
[2]http://dblp.uni-trier.de/
[3]http://www.cs.columbia.edu/ hgs/netbib/
[4]http://portal.acm.org/
[5]http://www.sciencedirect.com/
[6]http://citeseer.nj.nec.com/cs

new or overlapping. Basically, different information sources provides a different view for the the same data.

The system presented in this paper provides a neat unified view of the data. It tackles the problem of unification of same data from multiple sources. Unfortunately this poses the problem of having to rely on these sources for valid data. If they contain some wrong entries with missing information the system cannot do anything about it since it does not seek to gather information directly from the horse's mouth!

3.5 Conclusion

This chapter showed that both AIE and II are two vital technologies for the success of our research. AIE techniques provide the adequate capabilities necessary for learning new concepts while II allows our system to glue the learnt and harvested information together.

The different AIE technologies mentioned are a good representative of the most successful IE techniques available. They cover a wide range of methodologies and in Chapter 4 we will select one of these tools and utilise it in our methodology. With our approach we will show how such tools can be used in a more effective way.

With respect to II we decided to list only those systems which we believe are most relevant to our research. By doing so, we only gave an overview of II mainly because II is a very vast field, fragmented into several subfields and a complete review of the whole field would be beyond the scope of this thesis. For a more complete review of II technologies please refer to (2)(73).

In the coming chapters, we will see how both IE and II were used in our work. The usage of these technologies will not only produce better systems which are more

effective and efficient, but they will also add more value to the general user experience.

Chapter 4

Melita: A Semi-automatic Annotation Methodology

4.1 Introduction

The main goal behind the Semantic Web (SW) is to add meaning to the current web, therefore making it understandable by both humans and machines. Humans are already equipped for such a task, the knowledge they gathered throughout their lives, makes it possible for them to understand the content of different web pages without requiring any external help. When it comes to machines, it is a totally different story. They do not possess any background knowledge, therefore they cannot understand the information found on the web, since they cannot place it in any appropriate context. This poses a big constraint on the SW, this new web must somehow supply enough meanings (or semantics) to the data so that the machines can understand it while making sure it remains understandable by humans. If we manage to achieve this goal, we will have made it possible for knowledge to be managed automatically by the machines themselves, therefore reducing the need of human intervention.

There are several stages required before this goal is achieved. The first stages involve defining the infrastructure for the semantic web. New standards were defined

103

for representing knowledge starting from the very atomic levels where data is defined using the XML language and moving towards more elaborate and powerful representations, such as RDF. At this stage, the need was felt to abstract even further from the atomic elements and represent higher level concepts and relationships. To define these higher level structures for knowledge organisation (like ontologies, etc.), further standards were created such as OIL, DAML, DAML+OIL and finally OWL.

This is only half of the story. Having the adequate structures without the data is useless. Therefore, the next step in the SW is the population of these knowledge structures. There are several ways of populating these ontologies. The most basic way is to manually fill these knowledge structures with instances. But this is not very useful since we live in a dynamic world where things constantly change around us and probably those instances would have to change with time. Another way is to insert in the ontologies a reference to an instances rather than the actual instances. The process of associating parts of a document to parts of an ontology is normally referred to as annotation. In this way, if the instance changes slightly, there is no need of modifying all the ontologies where this instance appears. This also makes sense because in our world and even on the Internet, there exists no Oracle of Delphi that has the answers to all the possible questions thus we can never be sure of the validity of our data. Knowledge is by nature distributed and dynamic, and the most plausible scenario in the future (60) seems to be made up of several distributed ontologies which share concepts between them. The annotation task may seem trivial but in actual fact it is repetitive, time consuming and tedious to perform by humans. If we think about the current size and growth of the web (30), it is already an unmanageable process. If we re-dimension our expectations and try to annotate just the newly

created documents, it is still a slow time-consuming process that involves high costs.

Due to these problems, it is vital for the SW to produce methodologies that either help users during the annotation of these documents in a semi-automatic way or actually produce the annotations automatically. One of the most promising technologies in the Human Language Technologies (HLT) area, is without doubt Information Extraction (IE). IE is a technology used to identify important facts in a document automatically. The extracted facts can then be used to insert annotations in the document or to populate a knowledge base. The job performed by IE techniques fits perfect into the whole SW picture. IE can be used to support in a semi/automatic way knowledge identification and extraction from web documents (E.g. by highlighting the information in the documents). Also, when IE is combined with other techniques such as Machine Learning (ML), it can be used to port systems to new applications/domains without requiring any complex settings. This combination of IE and ML is normally called Adaptive IE (18)(7)(27).

There are already a number of tools that exploit AIE in order to support users while annotating documents. These include the MnM annotation tool (38) developed at the Open University (which uses both the UMass IE tools (1) and Sheffield's Amilcare (24)), as well as the Ontomat annotiser (64) developed at the University of Karlsruhe which is an implementation of the CREAM environment mentioned before. The approach which both MnM and Ontomat use to train the AIE algorithms is called active learning and works as follows:

1. the user annotates the document and the IE system learns how to reproduce those annotations.

2. when similar examples are encountered, the IE system automatically inserts

annotations based upon the previous cases.

3. the user finally checks the annotations.

4. the cycle starts all over again so that the algorithm is retrained with new corrected data.

It has been proven in (104) that this approach reduces the burden of manual annotation up to 80% in some cases. The advantages are quite obvious but unfortunately the interaction model between the users and the system which is implemented in both MnM and Ontomat is rather simplistic. Generally, the annotation process happens in a batch, i.e. the user annotates a batch of documents, when the batch is manually annotated, the IE algorithm is launched over that batch for training. Once trained, the IE system can be used over a new unseen batch of documents and the IE engine manages to recognise examples which are similar to the ones already seen in the training set.

Although we do not deny that these systems minimise the burdens of annotating documents, they are not effective enough and they do not exploit the full benefits of adaptive IE. In this chapter we propose a methodology[1] developed during the same period that MnM and Ontomat were being created, which implements an improved interaction model between the users and the system. It was originally designed to maximise effectiveness and reduce the burdens introduced by the annotation process. In so doing we would be moving from a system which interacts with the user to a system that collaborates with the user.

---

[1]This methodology was developed and refined by myself together with Professor Yorick Wilks, Professor Fabio Ciravegna and Dr Daniela Petrelli. The design and implementation of the system are exclusively mine.

The methodology to realise this interaction model is based around the Melita system. Melita is an ontology-based demonstrator for text annotation. As the basis for this system, we propose two user-centered criteria as measures of the appropriateness of this collaboration: timeliness and intrusiveness of the IE process. The first refers to the time lag between the moment in which annotations are inserted by the user and the moment in which they are learnt by the IE system. Basically Melita automatically calculates how timely is the system to learn from user annotations. In systems like MnM and Ontomat this happens sequentially in a batch. The Melita system implements an intelligent scheduling in order to keep timeliness to the minimum without increasing intrusiveness. The latter represents the level to which the system bothers the user, because for example it requires CPU (and therefore stops the user annotation activity) or because it suggests wrong annotations. It also refers to the several ways in which the IE system gives suggestions to the user to help reduce the burden of annotating tags.

4.2 Melita: A Semi-Automatic Annotation Methodology

In Melita, the annotation process is split into two main phases from the IES[2] point of view: (1) training and (2) active annotation with revision. In user terms the first corresponds to unassisted annotation, while the latter mainly requires correction of annotations proposed by the IES.

While the system is in training mode, the system behaves in a similar way to other annotation tools. In fact, at this stage, the IES is not contributing in any way

---

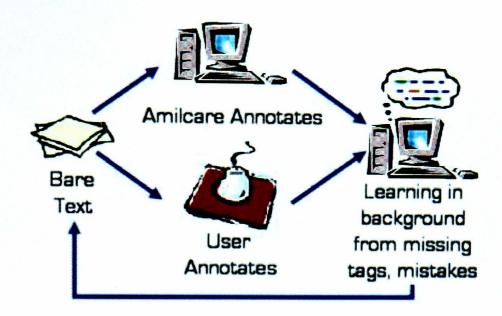[2]The IES used in Melita is Amilcare (http://nlp.shef.ac.uk/amilcare/).

Figure 4.1: Training with verification.

to the annotation process. If we take a closer look to what is actually happening in the background, we find that the system is not dormant. The IES uses the examples supplied by the user to silently learn and induce new rules (See Figure 4.1). This phase can be referred to as the bootstrapping phase whereby the user supplies some seed examples for an arbitrary document. The system learns new rules that cover those examples. As soon as the user annotates a new document, the system also annotates the document using the rules it learnt previously, and compares its results with those of the user. In this way, the system is capable of evaluating itself (when compared with the user). Missing annotations or mistakes are used by the learning algorithm to learn new rules and adjust existing ones. The cycle continues like that until the system reaches a sufficient level of accuracy (Refer to Section 4.3.1 for an explanation of how the sufficient level of accuracy is calculated).

Once this level is reached, the system moves over to the phase of active annotation with revision. In this phase Melita presents to the user a previously unseen document
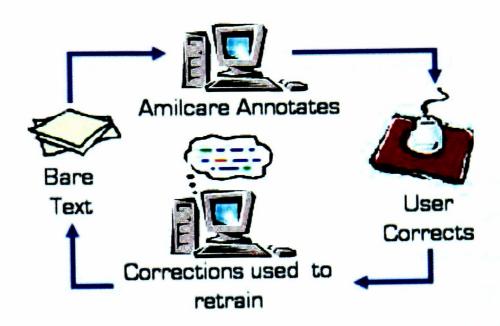
Figure 4.2: Active annotation with revision.

with annotations suggested by the system itself. At this stage, the user's task shifts from one of annotator to one of supervisor. In fact, the user is only expected to correct and integrate the suggested annotations (i.e. removing and adding annotations). When the document is corrected, these are sent back to the IES for retraining.

Figure 4.2 shows the phase where the system-user cooperation takes place. The system is annotating the documents for the user, while the user corrects any mistakes introduced by the system. In so doing the user would be implicitly giving back to the system important feedback regarding its annotation capabilities. These are then used by the system to learn new accurate rules and therefore improve its performance. The task of the user is also much lighter than before. Supervising and correcting the system is much easier and less error prone than looking for instances of a concept in a document. It is also less time consuming since the attention of the user is mainly focused towards the suggestions given by the system and the need of new manual annotations decreases when the accuracy of the IES increases.

4.2.1 Dealing with timeliness

In the Melita methodology mentioned above, timeliness[3] is only partially guaranteed, because the IES annotation capability always refers to rules learned by using the entire annotated corpus, less the last two documents. This means that the IES is not able to help when three similar documents are annotated in sequence. This is the price paid for learning in the background. From the user point of view the methodology is equivalent to training on batches of three texts, with all the disadvantages of batch training mentioned above (even if a batch of size three is quite small). Timeliness is a matter of perception from the user's side, not an absolute feature, therefore the only important matter - we believe - is that the users do not perceive it, rather than trying to avoid it at all costs. In this respect we start from the consideration that in many applications the order in which documents are annotated is random. Generally users adopt criteria such as date of creation or file name order in directories. In such cases it is possible to organise the annotation order so to avoid the possibility of presenting similar documents in sequence and therefore to hide the lack of timeliness. In order to implement such a feature we need a measure of similarity of texts from the annotation point of view. (41) tries to address this problem by proposing a committee-based sample selection approach. This involves examining the available corpus and selecting the most informative using some probabilistic classifiers. It avoids redundancy since the user is only annotating examples which convey new information to the IE engine.

In our approach, we use the IES to calculate such a measure. The user is initially asked to select the corpus of documents to annotate. These documents are then presented to the user for tagging one by one. As far as the user is concerned, they are

---

[3]Timeliness is the time lag between the moment in which annotations are inserted by the user and the moment in which they are learnt by the IE system.

presented in no particular order but in reality they have a smart ordering. Iterating through the selected corpus is not simply a matter of parsing the documents one by one. The point is that in reality, there is no need of examining all the training corpus. An IE engine can stop parsing the documents once all the concepts required are learnt. If we consider the training documents to be part of a document space in which all the training documents are represented, then the documents can be virtually clustered according to the number of patterns found by Amilcare during a test run. Melita then begins suggesting documents for tagging starting from those, which contain the least patterns. The reason being that these documents are more likely to have new patterns relevant to Amilcare, which were not yet learnt.

This process of smart ordering of documents cannot be initialised immediately and in fact it is started after two documents are tagged. To start immediately several other issues must be addressed such as how to select documents with the most relevant examples, and most important how can one denote relevance in new unseen domains? Since at the start of the training, no pattern is known by the training algorithm, therefore all the documents are arranged in no particular order. When a pattern is learnt by the IE algorithm, it is normally the case that other documents have similar patterns. Therefore if the other documents are tested with the IE engine, we can rate them according to the number of tags identified by the IE algorithm. These relevance ratings can be calculated only after the first document is tested and this is the reason why the process cannot be started immediately.

Since a rating of documents now exists, the next step is to choose the best document. The documents can be divided into 3 subcategories, in terms of whether they match any of the patterns so far proposed by the rule induction algorithm:

Documents without annotations Since we could be working in an open domain
we cannot be sure why these documents were not annotated. It might be the
case that these documents are irrelevant and therefore there are no patterns
to learn from them or the document is relevant but the learning algorithm did
not encounter any similar patterns in the already covered documents. Because
of the fact that we cannot know for sure which documents are relevant or not,
these documents are stored and will be used at a later stage.

Documents with partial annotations These documents are those, which have
new patterns not yet covered by the algorithm. To find out these documents,
the application must keep statistics of the average number of instances per
annotation suggested by the IE algorithm for an average document. These
statistics will help the application choose a document that has a high probabil-
ity of having examples containing new instances. A typical example is when the
learning algorithm returns from a document partial information covering only
few of the concepts. Since the document in question already contains some of
the patterns, there is a higher probability that it will contain other unknown
but relevant patterns. Therefore these documents must be sent for training in
order to allow the algorithm to learn all the missing concepts.

Documents having complete annotations These are documents which were an-
notated by the IE algorithm but were never verified by the user. The reason to
train the algorithm on these documents is to reinforce the IE algorithm. These
documents can be selected when the program enters Active Supervision (AS).
In AS the user is required to supervise or correct the output of the learning
algorithm rather than introduce new tags from scratch.

Once the documents are categorised, the best one can be chosen depending on the status of the application. In our model we always select in turn for annotation a completely uncovered document (i.e. a document from category 1) followed by a fairly covered document (document from category 2). In this way, the difference between successive documents is very likely and therefore the probability that similar documents are presented in turn within the batch of three (i.e. the blindness window of the system) is very low. Incidentally this strategy also tackles another major problem in annotation, i.e. user boredom. This is the major reason why the level of user productivity and effectiveness falls proportional to time. Presenting users with radically different documents should avoid the boredom that comes from coping with very similar documents in sequence. When the documents from the first two categories are annotated, those in category 3 are proposed for annotation (provided the learning algorithm did not already learn all the different patterns!). This makes sure that the documents which require most effort for the user are presented at the beginning of the session and those with least effort are presented towards the end of the session.

4.3 The methodology at work

Melita is an ontology-based demonstrator for text annotation. The goal of Melita is to demonstrator how it is possible to actively interact with the Information Extraction System (IES) in order to meet the requirements of timeliness and tunable pro-activity mentioned above. Melita's main control panel is shown in Figure 4.3. It is composed of two main areas:

1. The ontology (left) representing the annotations that can be inserted;
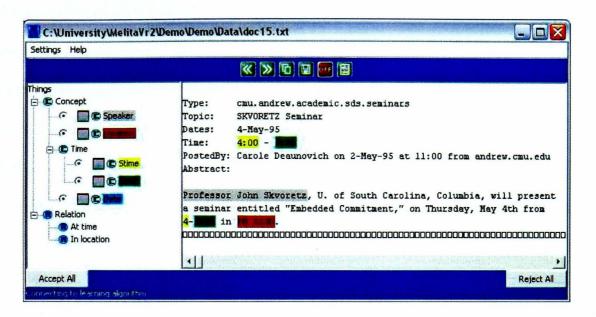
Figure 4.3: The Melita annotation interface

- Annotations are associated to concepts and relations.

- A specific color is associated to each node in the ontology (e.g. "speaker" is depicted in grey).

2. The document to be annotated (DOC panel) (center-right).

- Selecting the portion of text with the mouse and then clicking on the node in the ontology insert annotations.

- Inserted annotations are shown by turning the background of the annotated text portion to the color associated to the node in the hierarchy (e.g. the background of the portion of text representing a speaker becomes grey).

Melita does not differ in appearance from other annotation interfaces such as MnM or Ontomat. It differs in that it is possible to tune the IES so to provide the desired

level of pro-activity and in the possibility of selecting texts so as to provide timeliness in annotation learning.

The typical annotation cycle in Melita follows the two-phase cycle based on training and active supervision. Users may not be aware of the difference between the two phases. They will just notice that at some point the annotation system will start suggesting annotations directly in the DOC panel. In the initial phase, simple techniques that help the user identify instances in the document are used. Eg. If the user tagged a name in the top part of the document, a search is performed to identify other words, which match exactly that name in the rest of the document. These matches are than presented to the user so that he can decide weather they are correct or not.

Since Melita is ontology based, the different tags are defined in an ontology (See Figure 4.3). The component shows a hierarchical display of tags together with a small box on the left of every concept indicating the current level of accuracy reached by the learning algorithm for every tag. When the box is clicked, a level metre pops up showing the confidence of the IE algorithm to discover individual concepts. Internally, the system makes use of precision and recall[4] to calculate this level but in Melita, we just show one combined measure because a naïve user may not be acquainted with the meaning of precision and recall. Therefore considering the fact that it is normal to have high precision and low recall or vice-versa at the start of the training, this may disorient the user.

The user is also given the power to set at which level of accuracy the IES should start showing suggestions to the user. Different concepts can also have different levels

---

[4]The calculation which makes use of precision and recall is most commonly referred to as the F-measure.

of accuracy because particular tags may be more important than others therefore, a higher accuracy would be required. When the confidence level surpasses the level indicated by the user, the program utilises the feedback obtained from the IE engine on the given corpus to suggest occurrences of tags in the document. At this stage the role of the user changes from one of training the learning algorithm to one of actually supervising the algorithm. In fact the user will reach a point where he does not need to assist the IES any longer with new annotations but just correct the tags suggested by the learning algorithm. This is the concept of Active Supervision. The user is actively involved in supervising the work of the IE engine.

In Ontomat and MnM, the Learning stage is normally done in batches. In Melita, the system needs to constantly monitor the accuracy level of the algorithm so that it suggests annotations immediately when the accuracy goes beyond a user defined level. Because of this, the system does not train in batches but documents are processed almost immediately. The smart approach which Melita takes allow results to be obtained just in time when we need them and since all these evaluations are done in the background, the user does not even notice that the algorithm was being trained.

4.3.1 Intrusiveness

The fact that Melita manages to anticipate the user by highlighting suggestions immediately when the document is loaded is a very powerful feature. But this feature can also be very risky. By doing so, we would be taking away control from the user and placing it in the hands of the system. This is also referred to as intrusiveness, or rather the act of interfering without permission to do so. These interferences can be various, when the accuracy is high enough, any suggestion would be welcomed by users since it would save them a lot of work. But what about when the system
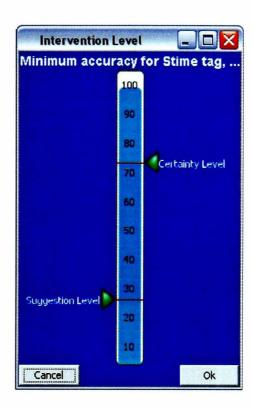
Figure 4.4: Intervention Level Setup

suggests annotations (during the active annotation with revision phase) when its suggestions are still unreliable? The system would be a nuisance for the users and would hinder them rather than help them.

To cater for this problem, Melita adopts a number of strategies. In a typical Melita session, once the first few iterations are complete, the program starts obtaining suggestions from the learning algorithm. These suggestions are stored and only shown to the user when and if the user decides to see them. This is done so that the program never interferes into the user's work without the user's explicit consent. Therefore with this system, we are empowering the user to calibrate the intrusiveness of the system in a very simple way. First of all the system has a suggestion button. Whenever a user disables it, no matter how many suggestions the system obtains, they are never shown to the user. If the user decides to view the suggestions all that

is needed is to enable this button and the suggestions are displayed immediately. In order to make things simpler for the user, the system calculates the level of expected accuracy[5] of the extraction process and uses that measure to either suggest to the user (if the accuracy level is low) or insert a tag in the document (if the accuracy is very high). This measure is possible because the rules in the IE system return an error score based on the training corpus.

The interface which allows the user to decide which rules should be used or not can be seen in Figure 4.4. It is made up of two knobs which the user can drag up or down along a percentage bar that lies in between the bars. The two knobs represent the suggestion level and the certainty level respectively. The value in the percentage bar is calculated using the accuracy measure returned by the algorithm. Basically, a low percentage in the bar indicate less accurate rules and a high percentage high accuracy rules. The two knobs, give an indication of which rules will be enabled.

- Annotations produced from rules below the suggestion level are never shown.

- Annotations produced from rules between the suggestion and certainty are enabled but are shown in the document as a colour bordered box. The fact that the box is just made up of a border and does not have a filling colour means that the program is not certain about its validity. If the suggestions are not validated by the user (by clicking on them), they are discarded before the document is saved!

- Annotations produced from rules above the suggestion level are enabled and showed using a coloured box. The fact that the box is fully coloured means that the program is fairly sure about the validity of the annotation. These

---

[5]Expected accuracy is calculated by finding the f-measure of the learning algorithm.

annotations don't require much validation from the user and they are inserted immediately in the document.

The purpose of this component is to allow the user to tune the level of precision and recall without actually knowing anything about them. The user is unaware that by moving the two knobs, he is calibrating the application. What he sees is tags appearing, disappearing or changing certainty level whenever a knob is moved. The reason for the creation of this component is to simplify the tuning of the precision and recall values individually for specific concepts.

This fine grained control per concept was created because the learner might need few examples in order to cover simple concepts and more examples for complex concepts. This allows the system to suggest annotations for simple concepts and let the user annotate the more complex ones. The advantage of this is that the user's attention is focused on the difficult tasks and the easy annotations are left to the system. The annotation interface must bridge the qualitative vision of users (e.g. a request to be more/less active or accurate) with the specific IES settings (e.g. change error thresholds) (28).

Another important aspect for intrusivity is processor time. Any IES is quite heavy on CPU usage when it comes to training. The IES can be so demanding at times that it could stop the user's activity. To solve this problem systems (MnM, Ontomat, etc.) often work in batch mode for training. Therefore the user annotates and then waits while the system is being trained. Normally this training time is delayed and scheduled for periods when the CPU is not used by the user such as during breaks, at night, etc. This approach unfortunately poses a problem of timeliness. The IES working in batch does not propose suggestions when required but instead does so

only after the system goes through a training session. This means that the user has to annotate a number of similar texts before the IES starts learning and a number of suggestions are proposed. Melita proposes a methodology to partially solve this problem by scheduling the learning phase in such a way that allows timeliness in the learning and limits to an absolute minimum the total amount of idle time imposed on the user.

The annotation process is normally made up of three phases, i.e. loading the document, annotating the document and saving the document with the annotations. If we examine these phases from the user's point of view we find that time wise, loading and saving are the less time consuming since they only take a negligible amount of time. Most of the user's time is spent annotating the documents. If we change our perspective and examine CPU time and effort, we find out that some of the computer's resources are used during the loading and saving phase while during the annotation phase, the load on the cpu is negligible. For this reason we believe that this is the right moment to train the IES in the background without the user noticing it. [6]

Therefore, we propose an interaction model made up of two parallel and asynchronous processes. The first process handles all the interaction between the user and the machine. It provides an annotation interface for the user and gives the user

---

[6] In theory, the system could start learning from the annotations as soon as they are inserted in the document even before they are saved to disk. This would make the process quicker but lets not forget that humans are prone to commit errors, so if an annotation is marked wrongly and the system starts learning immediately, it would be learning wrong annotations before the user has the time to correct it. If it is corrected by the user, the IES would have to modify the new rules back to how they were before the wrong example was learnt. Obviously this makes the whole interaction model between the user and machine much more complex.

the impression that he is using all the machine's processing power. The second process, works in the background and is used to train the learner as soon as new examples are supplied. This process uses a limited amount of processing power so that the user is not influenced due to lack of computing resources. What happens is the following, when the user loads a new document (lets call it document **N**), the IES applies the rules induced in the previous learning sessions[7] in order to extract information (either for suggesting annotations during active annotation or in order to silently test its accuracy during unassisted learning). Performing IE on one document is very fast and the user does not notice any delays. While the user is annotating document **N** the IES is learning from document **N-1**. This methodology reduces timeliness to an absolute minimum since the user is never idle. The time taken to train an IE system with new examples is normally faster than the time required to manually annotate a document. Therefore as soon as a document is annotated, the IES is ready, waiting to suggest new annotations for the next document.

4.3.2 Timeliness

The interface which gives information to the user regarding timeliness (See Figure 4.5) consists of the list of the documents in the corpus, rated according to the percentage of possible new patterns in every document. The document with the highest rating is the one that has some of the patterns but which also contains other patterns not covered by the learning algorithm and therefore it is the document suggested for training. The list of documents can also be sorted either by ratings or by name. To switch between orderings the user must select the title of either the column containing the file names or the other column. A user can also ignore the suggestions of the algorithm

---

[7]i.e. from document **1** to document **N-2** since document **N-1** is still waiting to be processed
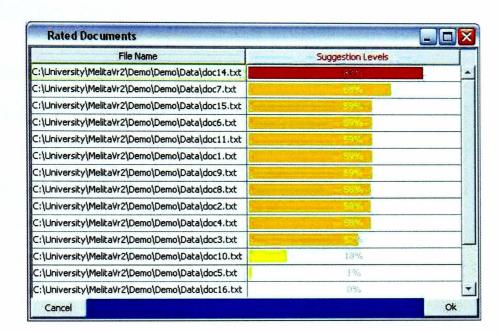
Figure 4.5: Documents rankings in Melita

and simply jump to a specific document by selecting the document from the list and pressing the "Ok" button.

4.4 Melita Evaluation

We performed a number of experiments both to demonstrate how effective the IE system is in learning concepts and to quantify its contribution to the annotation task. In the following sections we will highlight two specific tasks, the first is the CMU Seminar Announcements Task and the second is the PASTA Task. After these two experiments we will discuss alternative user oriented evaluations of the system. Finally we will peek into what we believe are the features which future annotation systems will have, followed by a general conclusion.

4.4.1 CMU Seminar Announcements Task

The first set of tests were performed on the CMU seminar announcements corpus[1]. This corpus is widely used in the IE field ((43), (17), (17), (47), (23) and (98)) to evaluate adaptive algorithms. In these experiments, we show how fast the IE system is able to converge to a status where it can really and effectively suggest. We also try to figure out its ability to suggest correctly and quantify its contribution to the annotation task.

The CMU seminar announcements corpus, consists of 485 documents[2] which were posted to an electronic bulletin board. Each document announces an upcoming seminar organised in the Department of Computer Science at Carnegie Mellon University. The documents contain semi-structured texts consisting of meta information (like sender, time, etc) which is quite structured and the announcement which is generally written using free text. The idea behind this domain is to train an intelligent agent capable of reading the announcements, extract the information from them and if it

---

[1]Originally prepared in (44)
[2]This corpus can be downloaded from http://www.isi.edu/ muslea/RISE/
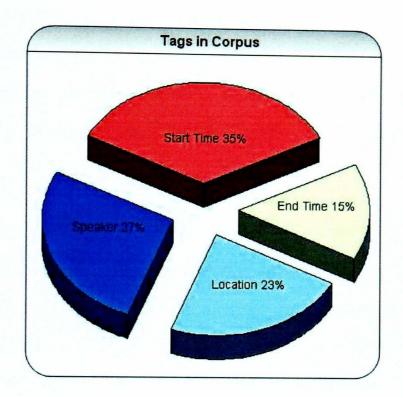
Figure 4.6: Distribution of the different tags found in the documents.

thinks they are interesting (based on some criteria predefined by the user) insert them directly in the user's electronic diary. In these documents, the following fields were manually annotated:

Speaker The full name (including any title) of the person giving the seminar.

Location The name of the room where the seminar is going to be held.

Start Time The starting time of the seminar.

End Time The finishing time of the seminar.

To better understand the task at hand, Figure 4.6 takes a look at the number of tags found in this corpus. It seems like the corpus is very rich in Start Time tags and less rich in End Time tags. It is generally the case that the more examples

we have the better (since we have more possibility to learn), but its not always the case. It is important to analyse another statistic related to the corpus. The graph in Figure 4.7 shows the number of distinct phrases containing a tag such as "**Start time: 12:00pm**", "**Seminar starting at 12:00**", etc; From this we can predict that:

Speaker will be quite difficult to learn. Intuitively, we know that a Speaker can be any sequence of words. A named entity recogniser can help spot names using linguistic cues (such as the titles Mr, Mrs, Dr, etc before a word) but this is not enough. Lets not forget that this task is not just a matter of spotting names but the system must identify the person who is going to give the seminar. From the graph we can see that there are around 491 distinct phrases containing names meaning that there is more than 1 new phrase (containing a name) per document. Even though there are 757 examples (around 27% of all tags) in the documents containing the Speaker tag, the fact that many of these examples are new and not repeated elsewhere, makes the whole task much harder.

Location too will be difficult to learn but definitely easier than the Speaker tag. This is very visible from the two graphs. First of all, the total number of examples in the corpus amounts to 643 or 23% of all the tags which is quite a reasonable representation. Secondly, the total number of distinct phrases is 243 which means that slightly more than $\frac{1}{3}$rd of the tags are new.

Start Time is a completely different story. The documents contain a total of 982 tags (or 35% of the total number of tags) and there are only around 151 distinct phrases. This means that training on 15% of all the documents (almost 75
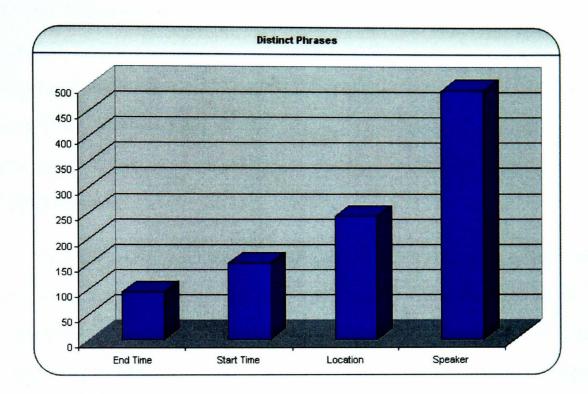
Figure 4.7: Number of distinct phrases (containing a tag) found in the corpus.

documents) is enough to learn this concept.

End Time is slightly more complex. The number of distinct phrases is very little, around 93 instances, but so is the representation of this concept in the corpus. In fact, this concept appears only around 433 times (i.e. 15% of all the tags). This means that there is a substantial number of documents where the End Time concept is not represented[3].

In the following experiments, the annotations in the corpus were used to simulate human annotation. The experiments were conducted as follows:

1. The corpus was randomly divided into two, a test corpus and a training corpus.

---

[3]Considering that the total number of tags amounts to less than the number of documents and there there are cases where more than one tag appears in a document

2. A cycle begins where we evaluate the contribution of the IE system at regular intervals during the tagging of the corpus. At every cycle, 5 new annotated documents from the training corpus (together with previously selected documents) are used to train the IE system. Therefore, we train the system by gradually adding 5 documents to the training corpus (i.e. 5, 10, 15, 20, 25, .., 150 documents).

3. As soon as the training finishes, we test on the test corpus. The ability to suggest on the test corpus was measured in terms of precision and recall. Recall represents here an approximation of the probability that the user receives a suggestion in tagging a new document. Precision represents the probability that such suggestion is correct.

4. The cycle continues like that until all the training documents are learnt.

The results are shown in Figure 4.8. The X-axis represents the number of documents provided for training while the Y-axis shows a percentage used to plot precision, recall and f-measure are presented. Although these results are similar to the experiments we published in (42), they have a number of important differences. The curves are more gradual yet they are smoother and manage to achieve higher precision and recall. This is due to a slight modification in the final version of the system than the one presented in that paper. Since the system always selected the best document for learning it was noticed that some concepts were being learned quicker than others. The reasons is that the "best" document for learning is the one with the most uncovered cases. If we take a look once again at Figure 4.6 and examine the distribution of the tags, we realise that since some concepts are represented more than others,

we would be discriminating against concepts which are not represented as much. Because of this, we modified slightly the selection algorithm so that we learn from the "best" document but we also keep into consideration the distribution of the tags. In our approach, every time we select a document, we do so by selecting the "best" document which represents a particular tag. Eg: The first document is the best document which represents the concept Speaker, the second one is the best document that represents the concept Location, etc. In this way all the tags are learnt at a steady pace. The annotations suggested by the system are more helpful for the user since they do not discriminate between concepts which are represented in the corpus more than others. The downside of this approach is that learning individual concepts is slightly slower because the system is trying to learn collectively all the concepts in the domain. Also, since the algorithm is learning from the "best" documents per concept, it is covering more unseen examples in the training phase therefore resulting in much higher levels of precision and recall. In fact, when the system is trained on just 50 documents the average F-measure for all concepts is beyond 75% (See Figure 4.9). The rise is constant and when 80 documents are trained, the system reaches almost 84% (See Figure 4.9) F-measure for all concepts. After that, the system reaches a sort of plateau where the improvement is very gradual.

As was predicted earlier, the maximum gain comes in annotating Start Time and End Time since they present quite regular fillers. Table 4.1 shows that after training on only 10 texts, the system is potentially able to propose 433 start times (out of 485), 373 are correct, 60 are wrong or partially wrong, leading to Precision=86 Recall=77. With 25 texts the recognition reaches P=94, R=68, with 50 P=97, R=80. The situation is quite similar for end time (although it takes more examples than start

Tag	Amount of Texts needed for training	Precision	Recall	F-measure
speaker	70	98	50	62
location	30	96	50	61
start time	10	86	77	80
end time	25	95	54	66

Table 4.1: Experimental results showing the number of training texts needed for reaching at least 75% precisions and 50% recall

time to converge), while it is more complex for speaker and location, where 70% and 80% f-measure respectively is reached only after about 80 texts. This is due to the fact that locations and speakers are much more difficult to learn than time expressions because they are much less regular.

The experiments show that the contribution of the IE system can be quite high. If we take a look at Table 4.1 we realise that with just limited training, reliable annotations can be achieved. This shows that the IE system contributes quite heavily to reduce the burden of manual annotation and that such reduction is particularly relevant. In the case of very regular concepts the benefits are immediate since the system starts suggesting from a very early phase. This allows the user to focus on annotation tasks which are more complex (e.g. speaker and location) while trivial and repetitive instances are handled by the algorithm. For complex tasks the system also contributes towards annotating the complex cases but some additional help is required from the user's side (Eg: in the case of speaker, twice the amount of documents required by the other concepts are needed by the system to reach 75% precision and 50% recall).

From these experiments, a number of issues arise. It is clear that the order in which documents are presented to the learner is important. Figure 4.9 shows the effect of
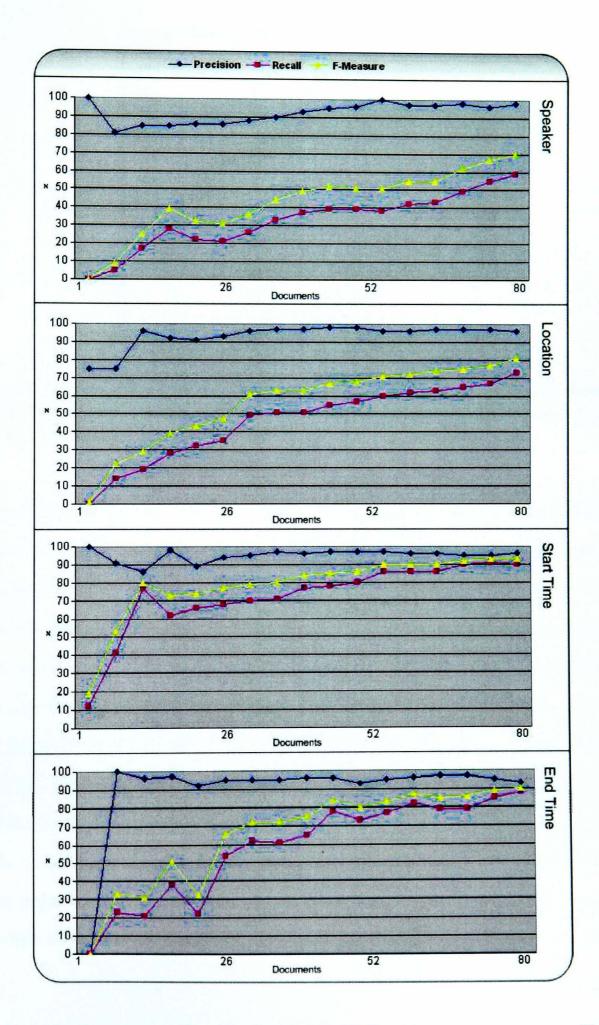
Figure 4.8: Experimental results for the CMU Seminar Announcements Task
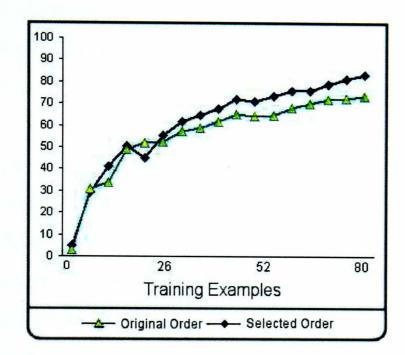
Figure 4.9: Experimental results showing a faster convergence when presenting ordered documents to the learning algorithm (The lines show the average F-measure for all the concepts)

selecting different document sets for training. When the IE system is presented with a random order of documents, its ability to converge to optimal results is limited for a number of reasons. First of all, the corpus contains similar texts therefore, even though the system is examining new documents, it is not examining new examples but examples which were already seen elsewhere. Secondly, the smart ordering of the documents presented to the learning algorithm allows the system to learn only from documents which contain new examples and therefore the coverage of the IE system increases drastically. This occurs because the value of the information found in the training corpus is much bigger and it allows a better convergence of the learner towards an optimal status. For example when 70 documents are annotated, the random order reaches a plateau (with very slow increases) having an F-measure of 71%, while the similarity-based selection provides the same F-measure with around

40 texts only. This means that users receive a much bigger help using a non-random order of document for training. This is consistent with the Active Learning theory, (104), and will require more investigation in the future for a better selection of the training corpus.

Another issues to consider is the way the user reacts to such help from the IE system. Since the system is capable of suggesting quite quickly, especially with concepts which are regular and therefore easy to learn, users could be tempted to rely on the IE system's suggestions only, avoiding any further action apart from correction. This could be dangerous for two reasons, both the quality of the annotations and the effectiveness of the system would decrease. Immediately from the start of the process, every newly annotated document is utilised for further training. The annotations are used to induce new rules and these rules are tested on the whole corpus in order to identify any false positives. If the user just corrects the learner without providing any new examples, the IE system will never learn any uncovered cases.

4.4.2 The PASTA task

The aim of the PASTA (Protein Active Site Template Acquisition)[4] project was to develop Natural Language Processing (NLP) techniques for the area of molecular biology. The main task was to extracting information on 3-dimensional protein structures from on-line scientific articles. As a result of this project, several PASTA corpora[5] were produced. To perform the PASTA experiments mentioned in this section, we downloaded two sets of abstracts made up of around 50 documents each. These documents are annotated with biological terminology information and were used in the

---

[4]http://www.dcs.shef.ac.uk/research/groups/nlp/pasta/

[5]The PASTA corpora can be downloaded from
http://www.dcs.shef.ac.uk/research/groups/nlp/pasta/results.html

PASTA project specifically for evaluation purposes.

The task we selected was to recognise in a set of 113 molecular biology related documents the following concepts:

- Species

- Residue

- Protein

- Site

- Region

- SecStruct

- SuperSecStruct

- Base

- Atom

- Non Protein

- Interaction

- QuaternStruct

We believe that this task can be considered a representative task for the Semantic Web especially since the molecular biology domain has attracted lots of attention in recent years. In our experiment the annotation in the corpus was used to simulate human annotation exactly as we did in the previous task. The documents in the two

corpora were grouped together and then randomly divided into two, a training corpora and a test corpora. A cycle begins where we evaluate the contribution of the IE system at regular intervals during the tagging of the corpus. At every cycle, 5 new annotated documents from the training corpus (together with previously selected documents) are used to train the IE system. Therefore, we train the system by gradually adding 5 documents to the training corpus (i.e. 5, 10, 15, 20, 25, .., 55 documents). As soon as the training finishes, we test on the test corpus. The ability to suggest on the test corpus was measured in terms of precision and recall. The cycle continues like that until all the training documents are learnt. Results are shown in Figures 4.10, 4.11, 4.12 and 4.13. On the X-axis the number of documents provided for training is shown. On the Y-axis precision, recall and f-measure are presented.

The IE task in the molecular biology domain is much more complex than in normal domains. If we take a look at the information to extract, we find elements like **CheY** and **Thr87**. These are typical names given to the structures we are trying to extract from the documents. It is immediately apparent that these names are very irregular therefore, the few cues we might use to identify these names must come from the context surrounding them. Even though the task is quite complex, the system is capable of proposing a significant amount of instances. After training on only 15 texts, the system is potentially able to propose more than half the instances for 3 out of the 12 concepts having an average precision of over 80% for the three of them. This is possible because of Amilcare's ability to generalize over both the text context and the filler where possible. Since most of these concepts are names of biological structures, some of them follow the typical format of names i.e. they start with a capital letter followed by lowercase letters. In this case we might be able to exploit

Tag	Amount of Texts needed for training	Precision	Recall	F-measure
Species	25	100	50	62
Residue	20	95	55	66
Protein	25	95	50	62
Site	40	100	50	62
Region	35	100	58	70
SecStruct	25	100	50	62
SuperSecStruct	20	100	60	71
Base	15	100	61	73
Atom	15	90	50	61
Non Protein	35	98	53	65
Interaction	15	83	55	64
QuaternStruct	25	100	50	62

Table 4.2: Experimental results showing the amount of training texts needed for reaching at least 75% precisions and 50% recall

a Named Entity Recogniser such as Annie (www.gate.ac.uk). Even if the algorithm does that, please note that the recognition of proteins, residues, secstruct, etc is not a simple Named Entity Recognition task. The system must not only recognise the names, but it must also assign it to the correct type. There are many names in the documents, therefore this implies the ability to recognise the context in which the names appear. The situation is more complex for other fields such as site, where 80% F-measure is reached after 50 texts. These annotations are much more difficult to learn than expressions whose filler are either very regular (e.g. interaction) or can be listed in a gazetteer (e.g. base), because their regularity is much less direct.

In the case of the PASTA task, our experiments show that it is possible to move from bootstrapping to active annotation after annotating a very limited amount of texts. In Table 4.2 we show the amount of training needed for moving to active annotation for each type of information, given a minimum user requirement of 75%
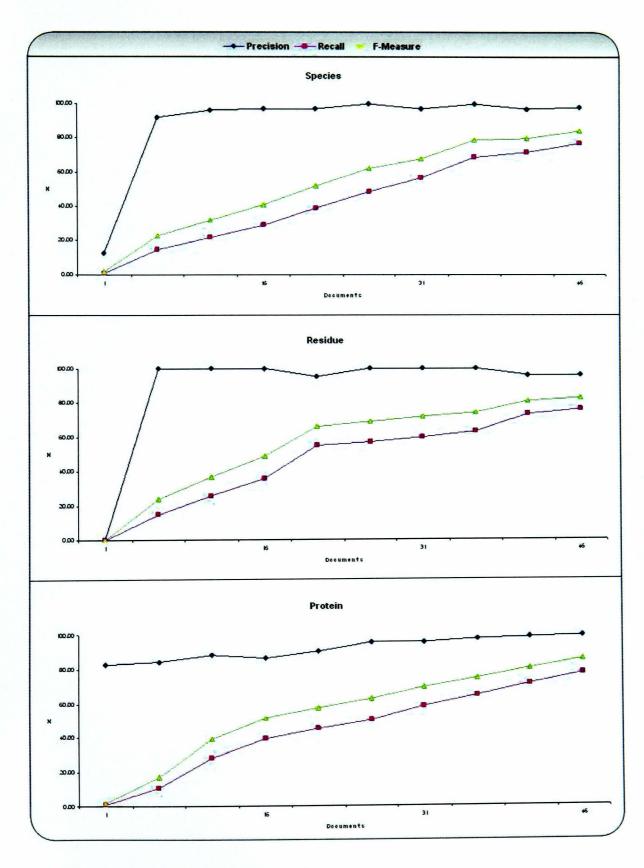
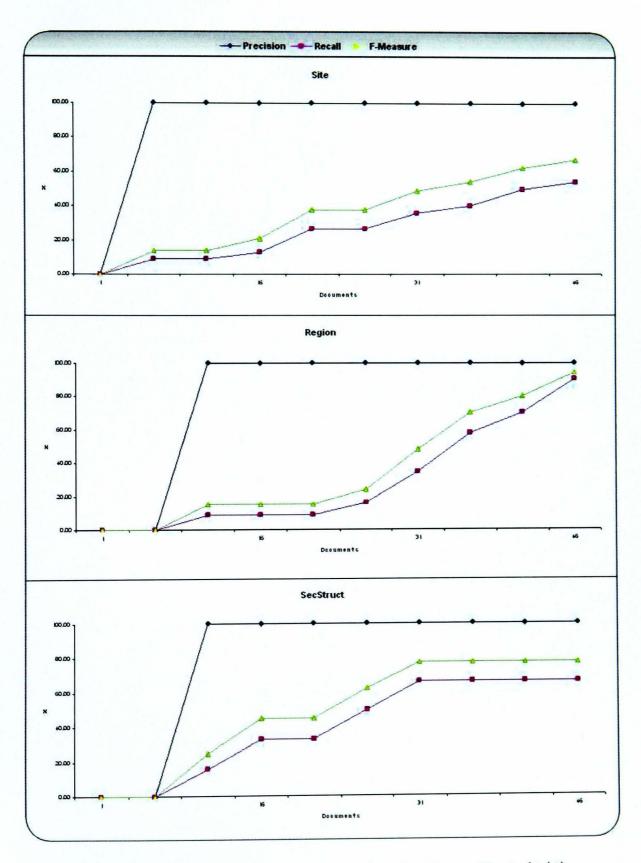Figure 4.10: Experimental results for the Pasta Task (1/4)

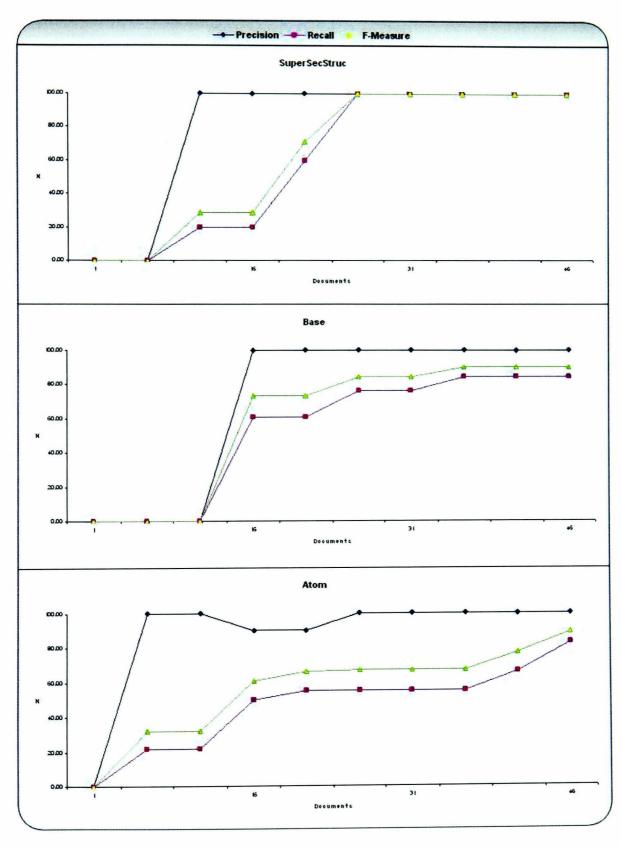Figure 4.11: Experimental results for the Pasta Task (2/4)

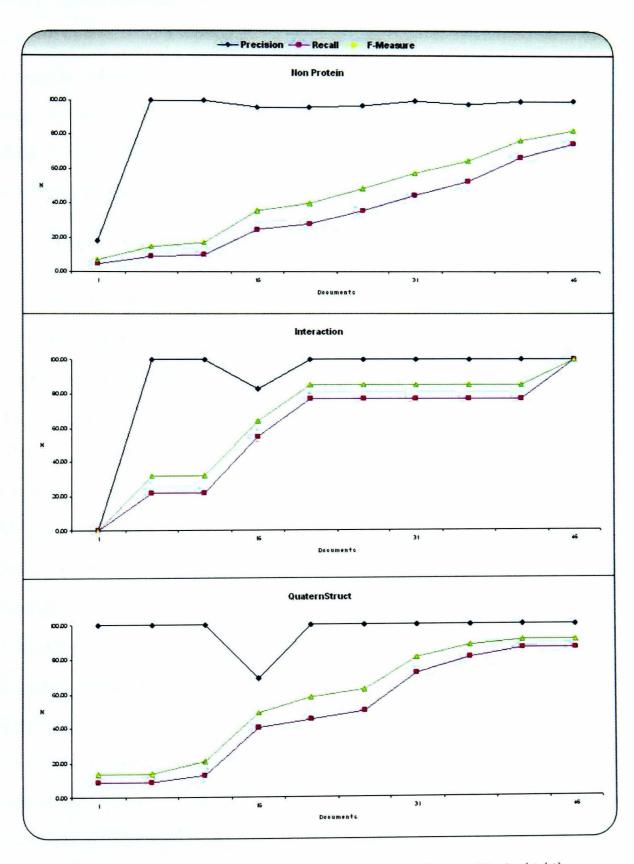Figure 4.12: Experimental results for the Pasta Task (3/4)

Figure 4.13: Experimental results for the Pasta Task (4/4)

precision. This shows that the IES contribution heavily reduces the burden of manual annotation and that such reduction is particularly relevant and immediate in case of quite regular information (e.g., base). In user terms this means that it is possible to focus the activity on annotating more complex pieces of information (e.g. site), avoiding being bothered with easy and repetitive ones. With some more training cases the IES is also able to contribute in annotating the complex cases.

4.4.3 User Oriented Evaluation

During these past years, Melita was distributed to several research groups, to some commercial companies and it was also used by students in several summer schools. The main reason for doing this was because we wanted the users to evaluate our system not only on test domains, but also on real life applications. The result from the Melita distribution was that we managed to obtain a lot of feedback which we then used to adjust and enhance our system.

To make life simple for the users, the first thing we did was to create a user manual. Secondly, whenever we were requested to send a copy of the system, if possible we used to go and give a demonstration of how the system works, if not, we used to send an animated video clip instructing the users how to utilise Melita. There were never any problems with regards to usability of the system since it was designed to be simple to use.

The general comment we got back from the users was that the system indeed manages to offer them a better annotation experience. The learning curve was very shallow and a typical user manages to master the system very quickly. The interface is quite straightforward, no complex numbers which bother the user with IE statistics are shown (in fact these measures are masked using graphical components which

everyone can understand), there are very few buttons and they all have a well defined purpose. Thus, the user immediately feels comfortable with the system. When the annotations become complex, and the different annotations are cluttered on top of each other, the system is capable of hiding the annotations which are not being used so that the user's attention is focused only on the task at hand. This feature was appreciated a lot by the users since it makes complex tasks manageable. Melita's compliance with well established standards also helped new users to configure the system to use new domains without much difficulty. In fact, since Melita already accepts ontologies in DAML format, the only configuration necessary to set up a domain was to load the ontology.

With regards to functionality, Melita also offered a number of simple yet powerful features. The annotation process only involves, selecting a concept and highlighting the instances of that concept in the document. Whenever an instance already highlighted by the user is encountered somewhere else in the document, it is automatically highlighted for the user. The users reported three main benefits of this feature:

Missing/Wrong annotations. It was noticed that throughout the annotation session, the performance of the user starts declining and the time taken to annotate a document becomes inversely proportional to the amount of annotation errors. In simple terms, the more time the user spends annotating a group of documents, the more he is error prone. Thus, using this feature, the system takes care of annotating instances already encountered and the user's attention is only focused on discovering new ones.

Avoid duplication. Since duplicate instances which were already encountered by the system will be highlighted automatically, the user is relieved from the tedious

task of re-annotating the same instances.

Consistency. This feature avoids the problem of inconsistent annotations whereby same elements are annotated differently in different documents. E.g. should we annotate `Mr Thomas Smith` or just `Thomas Smith` ?

The highlights introduced automatically by using the learning algorithm, were warmly welcomed by most users. Unfortunately, this feature raised a hot intrusiveness issue. Even though the system allows the user to tune the intrusiveness, some user felt that the system was suggesting annotations too soon and thus disrupt their work since their focus is shifted towards verifying the system's suggestions. Because of this, the system was enhanced to meet the user's requirements and a new button was added which gives the users the power to hide or show the suggested annotations when they want. Thus, the system does not present to the user the new suggestions immediately but only when the user requests them.

The smart ordering in Melita was also very useful during the annotation phase since the users were not bothered with highlighting similar documents. This meant that each document, presented to the users for annotation, was different from the previous one. The result was that the annotation task was more intellectually challenging for the user and therefore reduced the boredom associated with the annotation task.

Finally, since Melita was used in many group activities (such as summer schools, etc.), its collaborative annotation features became very useful. In such cases, the Melita server was installed somewhere centrally and users collaborated in annotating the same domain. Whenever a user annotated a document, the annotations found in that document were learnt by the IE engine and utilised to annotate the documents of the other users as well. This obviously led to a much faster successful completion

of the annotation task at hand.

Our focus was on the users from the times when we were still designing Melita. Overall, the feedback we got back from them was very positive and could be used in the future to do some more formal and quantitative analysis. Through this feedback, we identified two kind of analysis which can be performed:

Features comparison refers to the different features offered by the tool such as smart ordering, simple suggestions, IE suggestions, etc. In such an experiment, a group of users (who must all be confident with the annotation process but preferably not confident with Melita) are seated in a lab. Each one will be asked to annotate several test sets and to do so, they have a specified time limit. The configuration of the tool on each machine will be different i.e. having some of the features switched off. Melita writes all the events happening to a log file together with a time stamp, thus, at the end of the session, this log file can be fed to an automated process which would analyse the different users and configurations, and produce an analysis of how much the different features (or a mixture of them) contributed to the annotation process in terms of speed, error avoidance, etc.

Tool comparison refers to the different annotation tools freely available on the market such as MnM, S-CREAM, etc. This comparison would be a little bit more tricky than the one before, since all these tools do not provide a log file which can be analysed by an automated process, thus, all the evaluation must be conducted manually. In this case, we could organise a lab having several users all using different annotation tools. They would be given a common task, basically to annotate a collection of documents, together with a time limit. At

the end of the session, we can analyse how many documents were annotated using each tool and also we could compare the annotated documents in order to identify which tool produced more qualitative annotations (i.e. those which are less error prone and consistent).

In both experiments, more than one user will be given the same configuration for the experiment and then, an average of their performance will be taken. This would avoid the possibility of having misleading evaluation results.

Although we don't deny that such an evaluation would be very useful, we had many one-to-one contacts with different users and the feedback we got encouraged us to move our research forward by developing new tools which reduce further the burden on the user when it comes to annotate new documents. This was our main motivation when we decided to develop Armadillo, the system which will be described in the next chapter.

4.5 Future Development: Adapting to different user groups

The methodology as a whole is quite generic and with it, we have targeted the most common users, but the reality is that there are several categories of users all with different backgrounds and needs. Because of this, we divided the users into three distinct categories and a potential future tool could be designed to cater for their individual needs. We also acknowledge the fact that users may not fall in exactly these categories but somewhere in between. Therefore the system should make sure that a user from one category can use tools designed for other categories. The main categories are Naïve Users, Application Experts and IE Experts.
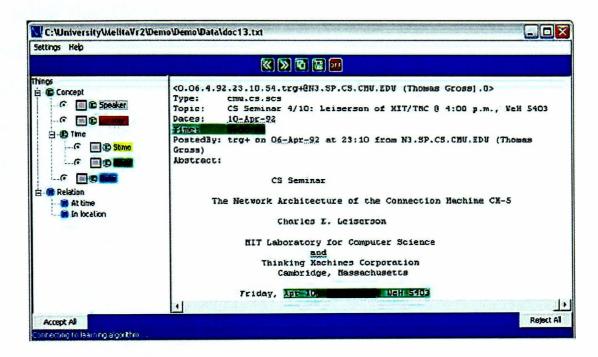
Figure 4.14: Naïve users Interface

4.5.1 Naïve Users

These users are familiar with their domain but have limited knowledge when it comes to computing. In order to set up the system they only need to specify an ontology and a corpus of documents. Apart from this, all that is required from the user to use the system is the ability to highlight concepts in the document according to concepts in the ontology. The system will also offer advanced features disguised as simple widgets like the component that tunes precision and recall (See Figure 4.4). By using this component the user will see tags being updated in real time according to the movement of the knob and the calibration of the component stops when the results are satisfactory.

A potential contribution of this kind of user is domain knowledge. In order to exploit this, the system will highlight words found around the concept that are part of the rules induced by the IE engine. These words will be highlighted using a slightly

lighter colour than the highlight of the main concept (See Figure 4.14), to show they are used to help identify the concept. The user will be able to remove or add such highlights. By doing so the user will be unconsciously guiding the algorithm to use certain cue words which are good at identifying the current concept. Therefore the algorithm will induce rules faster because it can use heuristics indirectly provided by the user. It will also converge faster because it is being guided by the domain knowledge of the user.

4.5.2 Application Experts

The people in this category are at a level between naïve and expert users. They are capable of tuning the application but do not have the expertise of an IE expert. Our tool will allow them to have access to advanced features such as adjusting IE parameters, setting the precision and recall levels explicitly, using the IE system interface and accessing other internal settings.

Due to the fact that this kind of user will have limited knowledge about the nature of information but also considering that their role is to maximise the potential of the IE engine, it is imperative that they are allowed to tweak rules in the simplest way possible. The system provides an abstraction for rules in the document being edited. It highlights the words which form part of a rule using a slightly lighter colour than the highlight of the concept they are associated with. When a word is selected by the user, a percentage is shown indicating the level of generalisation of that word used by the rule. 0% indicate that a very specific level is used (i.e. the word is explicitly part of the rule) and 100% indicate the most generic level. By sliding the percentage bar the user will be able to see the effect of those changes in real time through components which graphically show precision, recall, f-measure and error levels of
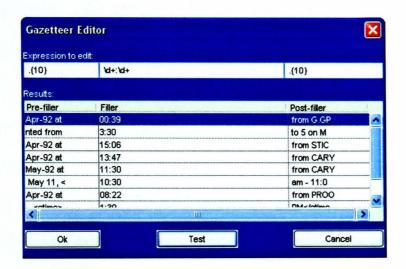
Figure 4.15: A rule editor for regular expressions based rules

the new rule. The highlights are not fixed and the user can add or remove any of them. This tweaking of rules will allow the user to change rules without the need of understanding linguistic properties. Current rule editors for IE like the Amilcare Rule Editor, allow users to change rules but an implicit linguistic knowledge is an absolute requirement, therefore restricting the use of the tool to IE experts.

4.5.3 IE Experts

Our final user is the IE expert who knows how to control an IE engine and wishes to extract the maximum power from the AIE system. All the benefits offered to the other type of users will be available but this user will require more sophisticated tools. Therefore, the system has a fully fledged rule editing environment.

Using this environment a user can view a list of rules together with statistics about every rule. In this view rules can be compared simultaneously. This is done in order to help the user restructure groups of rules. For example, it may be the case that a group of similar rules can be compressed together in one rule. This view will

also enable easy browsing and selection of individual rules. Once a rule is identified as requiring change or even if the user would like to create a new rule, the rule is opened in the rule editor (See Figure 4.15). This program presents to the user several facilities in order to allow him to develop the best rule possible. He has the faculty to change individual properties of a rule and also insert new ones. The changes on the rule can be tested immediately in real time and all the examples in the text where the rule applies are presented to the user. Based on previously annotated documents the system displays positive and negative examples covered by the rules. The system will also display examples of where the rule fires in the test corpus (which is untagged). At this stage the user can separate the positive from the negative results and using this new knowledge the system induces a new rule which is presented to the user for verification. The cycle continues until the user is satisfied with the new rule.

What is actually happening in the rule editing environment is that we are again using our methodology but this time at a deeper level in order to help the user create handmade rules quickly. The system will be guiding the user towards creating effective rules both by suggesting new rules and by providing important statistical results.

4.6 Conclusion

IES can strongly support users in the annotation task, alleviating users from a big deal of the annotation burden. Our experiments show that such help is particular strong and immediate for repetitive or regular cases, allowing the focusing of the expensive and time-consuming user activity on more complex cases. Despite these positive results, we claim that the simple quantitative support is not enough. An

interaction methodology between annotation interface, user and IES is necessary in order to reduce intrusivity and maintain timeliness of support. The methodology implemented in the Melita tool addresses the following concerns:

1. It operates in the usual user environment without imposing particular requirements on the annotation interface used to train the IES (reduced intrusiveness).

2. It maximises the cooperation between user and IES: users insert annotations in texts as part of their normal work and at the same time they train the IES. The IES in turn simplifies the user work by inserting annotations similar to those inserted by the user in other documents; this collaboration is made timely and effective by the fact that the IES is retrained after each document annotation.

3. The modality in which the IES system suggests new annotations is fully tuneable and therefore easily adaptable to the specific user needs/preferences (intrusiveness is taken under control).

4. It allows timely training of the IES without disrupting the user pace with learning sessions consuming a large amount of CPU time (and therefore either stopping or slowing down the annotation process).

Unfortunately, there are still a number of open issues in our methodology such as the effect on the user, of excellent IES performances after a small amount of annotation. For example when P=86, R=77 is reached after only 10 texts (as for Start Time in the CMU seminar announcements task), users could be tempted to rely on the IES suggestions only, avoiding any further action apart from correction. This would be bad not only for the quality of document annotation, but also for the IES effectiveness. As a matter of fact, each new annotated document is used for

further training. Rules are developed using existing annotations. They are tested on the whole corpus to check against false positives (e.g. the rest of the corpus is considered a set of negative examples). A corpus with a relevant number of missing annotations provides a relevant number of (false) negative examples that disorients the learner, degrading its effectiveness and therefore producing worse future annotation. The entire dimension of the problem is still to be analysed.

Even though there are these issues, Melita is still a huge improvement over the existing systems. In the coming chapter, we will show how we can build further on top of the methodology presented in Melita by making a small modification to the system. The resultant effect will be, a generic and fully automated approach which can be used over the Web to insert sematic annotations in web documents.

Chapter 5

Armadillo: An Automated Annotation Methodology

5.1 Introduction

The Semantic Web (SW) needs semantically-based document annotation[1] to both enable better document retrieval and empower semantically-aware agents. Unfortunately, although the Melita methodology presented in Chapter 4 can be applied to a wide number of domains, it is not suitable for the generic Semantic Web task.

Even though Melita changes the annotation task from a completely manual (such as (65)) to a semi-automatic task (26), it is still based on human centered annotation. Whenever a user is involved in the annotation process, although the methodology relieves some of the annotation burdens from the user, the process is still difficult, time consuming and expensive. Apart from this, considering that the SW is such a huge domain, convincing millions of users to annotate documents is almost impossible since it would require an ongoing world-wide effort of gigantic proportions. If for a second we assume that this task can be achieved, we are still faced with a number of

---

[1]Semantic annotation is the process of inserting tags in the document, whose purpose is to assign semantics to the text between the opening and closing tags.

open issues.

In this methodology, annotation is meant mainly to be statically associated to (and saved within) the documents. Static annotation associated to a document can:

1. be incomplete or incorrect when the creator is not skilled enough;

2. become obsolete, i.e. not be aligned with page updates;

3. be irrelevant for some use(r)s: a page in a florist web site can be annotated with shop-related annotations, but some users would rather prefer to find annotations related to flowers.

Different annotation can be imposed on a document using different ontologies. An ontology is required because it describes concepts and relationships that occur in a very restricted view of the real world, basically it describes the domain in which we are working. In the future, most of the annotation is likely to be associated by Web actors other than the page's owner, exactly like nowadays' search engines produce indexes without modifying the code of the page. Producing methodologies for automatic annotation of pages with no or minimal user intervention becomes therefore important: the initial annotation associated to the document loses its importance because at any time it is possible to automatically (re)annotate the document and to store the annotation in a separate database or ontology. In the future Semantic Web, automatic annotation systems might become as important as indexing systems are nowadays for search engines.

Therefore, we propose a methodology that learns how to annotate semantically-consistent portions of the Web by, extracting and integrating information from different sources. All the annotation is produced automatically with no user intervention,

apart some corrections the users might want to perform. The methodology has been fully designed and implemented by myself in Armadillo[2], a system for unsupervised information extraction and integration from large collections of documents.

The natural application of such methodology is the Web, but large companies' information systems are also an option. In this chapter we will focus on the Web, and in particular we will take a look at the task of mining Computer Science Departments websites. All the process is based on integrating information from different sources in order to provide some seed annotations. This will then bootstrap learning which in turn will provide more annotations and so on. In synthesis we start with a simple methodology which requires limited annotation, and move on to produce further annotation to train more complex modules.

In the next section we will present the generic architecture that we have used to build the application. Then we will describe the CS Department task. It will be shown how the information from different sources is integrated in order to learn to annotate the desired information.

5.2 Armadillo: A Generic Architecture for Automatic Annotation

The Armadillo methodology is centred around the idea of exploiting the redundancy of the web to bootstrap IE learning. This idea is not new since it was proposed by Brin (15) and Mitchell (87). The difference with our approach is the way in which learning is bootstrapped. Brin uses user-defined examples, while Mitchell uses generic patterns that work independently of the site or the page being analysed. We use both

---

[2]http://www.aktors.org/technologies/Armadillo/
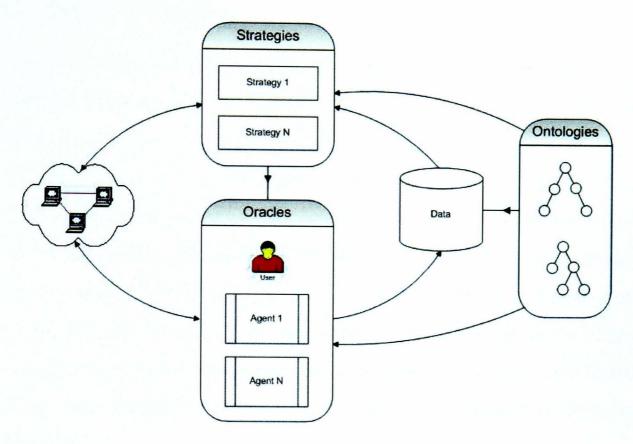
Figure 5.1: The Armadillo Methodology.

the above, but in addition, we exploit the redundancy of information and integrate information extracted from different sources with different levels of complexity.

Since there can be many potential applications of the proposed technology, we decided to make the Architecture as generic as possible. In order to do so, the Architecture had to be portable and scalable. This was achieved by making use of a simple methodology that requires no special configurations built around a distributed web service architecture. The methodology is made up of four main items, these include a set of Strategies, a group of Oracles, a set of Ontologies and a Database (See Figure 5.1).

5.2.1 Set of Strategies

Strategies are modules capable of extracting information from a given document using very simple techniques. Each strategy takes as input a document, performs a simple extraction function over that document and returns a typed output.

The input document is not restricted to any particular text type and can range from free text to structured text. The limitation is only imposed by the processor involved in pre-analysing the particular document. In fact, there is nothing stopping the input document from being a picture or a piece of music. The module itself is responsible for implementing or accessing these preprocessors. Since the system is built around a distributed architecture, the preprocessor (if needed) could be either a process within the module or even a process running somewhere over the network like a web service.

The extraction functions found in the strategies do not use any complex Artificial Intelligence (A.I.) techniques but rather weak strategies such as simple pattern matching routines. The idea is that whenever weak strategies are combined together, they manage to produce stronger strategies. The process does not have to be something perfect, an error rate is tolerated at this stage since the validity of the data will be further confirmed by the Oracles.

The output from every module is a set of instances belonging to a particular concept which is described in one of the ontologies used by the system. These instances are stored in the central database and are typed according to the name of the concept which they belong to.

To better illustrate the role of a strategy, imagine we need to extract the potential names of people found in a particular document. For this task, we do not need any

particular preprocessor. We just need a good heuristic to extract the names. A very simple yet highly effective heuristic would be to extract all the bigrams[3] containing words starting with a capital letter. This works pretty well and manages to return bigrams like "Alexiei Dingli", "Yorick Wilks", etc. One can argue that this approach would probably return some garbage as well like the words "The System". This is true, but this problem is solvable in two ways. First of all, there is nothing stopping a module from having a postprocessing procedure inside it used to filter away garbage obtained by the simple approaches. In fact, the module used as example in this paragraph is part of the current implementation of the Armadillo system and it does have a post processing procedure which filters out bigrams that contain stopwords[4] inside them. This would filter out wrong instances such as "The System" since the word "The" is a stopword. Secondly, we should keep in mind that any instance returned by this module must be verified by an Oracle before being inserted in the main database.

In synthesis a strategy is a module that allows us to implement very simple yet extremely fast strategies in order to extract information. They do not have to be highly sophisticated tasks and not necessary with high precision. The most important thing is that these strategies provide some seed examples which are used by the Oracles to discovery of other instances.

5.2.2   Group of Oracles

An Oracle is an entity which can be real or artificial, that possesses some knowledge about the current domain. To better understand what Oracles do, it might be useful

---

[3]A bigram is a pair of adjacent words.

[4]A stopword is a word that is very common inside a textual document (e.g. prepositions and articles).

to think about them as filters. These filters accept as input some data of a particular type found in the ontology and return data of the same type, but the data they return is validated. The validity of this data is guaranteed by the Oracle. This adds a certain degree of accountability to the system since an Oracle is responsible for the data it validates. Therefore, if an item of data is wrong and it was validated by a particular Oracle, we could pinpoint exactly which Oracle validated the data and take appropriate corrective actions. These actions include adjusting the validation mechanism of the Oracle or even excluding it from future validations. The exclusion of one Oracle is normally not a big loss since a system has different Oracles performing the same validation using different methods. Having redundant Oracles is very important if we want to exploit the redundancy of the web to the full. Since the same information is available from different reliable sources, it is vital for the system to have an Oracle for each and every source. The combination of these sources will thus produce very reliable data since it is not up to one Oracle to decide if an instance is valid or not but rather to a committee of Oracles (similar to having a panel of experts to evaluate the data).

An Oracle performs another task apart from the filtering, if the Oracle posses some additional information on the subject being analysed which is not part of the central database, it adds it to the database. This information is reliable since it is supplied by the Oracle.

To summarise, an Oracle filters away information that is marked as being an instance of a particular domain but which in reality is not (according to the Oracle) and adds any additional reliable data it posses. There can be different types of Oracles such as humans, gazetteers, web resources and learning algorithms. The list is not

an exhaustive one but highlights the most commonly used Oracles.

Humans are the best kind of Oracles we can use. They contain a huge amount
of information and they are excellent information processing systems. The
problem with humans is that the whole scope of this system is exactly to spare
them from doing the tedious job of finding instances in documents. Our strategy
is to make use of humans only if we cannot find any other Oracle to do the job
and use them only for the starting phase in order to initialise the whole process
with a small set of high precision data. Then, the remaining tasks should be
preformed by the other kinds of Oracles. This does not mean that the role of
the user stops there, at any stage the user can see what the different strategies
and Oracles are producing, and review their data if necessary.

Gazetteers are lists of elements which belong to the same class. These lists can
contain anything like lists of names, countries, currencies, etc. Each list is
associated with a concept found in one or more ontologies. If the output ob-
tained from one of the strategies is found in one of the lists, then that element
is confirmed as being an instance of the particular concept represented by the
list. The element would have both a time stamp and a signature signifying
that the information was confirmed by this particular Oracle. Lets assume
for a second that one of the strategies returns the words "United Kingdom". A
search is performed through the lists to identify whether this word is an instance
that occurs in any of the available lists. In this example the phrase "United
Kingdom" was found in the gazetteer called countries. This gazetteer is itself
attached to a concept in one of the ontologies used called *country*. Therefore,
the system assumes that "United Kingdom" is equivalent to the instance found

in the countries gazetteer and thus, it is semantically typed as being a country. Once we have semantically typed data which is verified by an Oracle, we can insert it in the database. Apart from Gazetteers, an Oracle could also exploit a Knowledge base containing a store of knowledge about a domain represented in machine-processable form. The data may include rules (in which case the knowledge base may be considered a rule base), facts, or other representations.

Web resources include any reliable list or database found over the web which can be used to verify wether an instance is part of a particular class or not. The information must be very reliable and up to date since at this stage, a minor error rate is not allowed if we want the data to be reliable. The task of querying these web resources, it is not always straightforward. First of all, if the web resource is accessible through a web service, the system can easily access the web service by making use of standard techniques. If no web service is available, a strategy must be defined by the user in order to instruct the system which information to extract from the web page. Luckily for us, these pages are normally front ends to databases. Therefore, since their content is generated on the fly using some program, the layout of the page would be very regular. In this case, the user is asked to mark a couple of examples in the documents and the system learns how to extract these instances by using an underlying IE engine. The learning algorithm used is based on simple wrapping techniques and manages to achieve very good results since the web pages are very regular. These resources are not very different from a gazetteer, the main difference being that while a gazetteer is generally a static list, a web resource has dynamic data which is continuously being updated by a third party.

Learning algorithms include technologies such as ML and IE tools, basically any classifier available which can identify whether an element is part of a particular class or not. These algorithms are much more sophisticated than what we have seen so far. They specialise on extracting information from individual documents which are not regular and therefore, learning a common wrapper as we have seen before is impossible. These algorithms do not even need any training by humans. They typically get a page, partially annotate it with the instances which are available in the database, they learn from those instances and extract information from the same page. The cycle continues like that until no more instances can be learnt from the page. The main difference from these classifiers and the ones mentioned before in the strategies is that they must have high precision since at this stage, we are verifying the data and no longer harvesting the information.

5.2.3 Ontologies

An ontology is a formal, explicit specification of a shared conceptualisation of a domain of interest (60). In simpler terms, the ontology specifies the concepts found in a domain[5] together with some existing relations between those concepts. Ontologies are important for the system to work correct because they are the glue that makes it possible for the data produced by the system to be related together. Apart from this, the different modules are semantically typed meaning that whatever they return must always be an instance of one of the concepts in the ontologies. To understand better the role of the ontology, imagine we have a module which returns the title of a book called "Animal Farm" and an author called "George Orwell" from a web

---

[5] A domain is a subset of the real world

page found in an online book store. Although for us it is quite easy to figure out that the two pieces of data are related i.e. an author writes a book and that book has a title. For the system, there is no way in which this can be achieved unless it is explicitly stated somewhere. Ontologies are the places where these associations between concepts are expressed. Using the ontology and the data gathered from the other modules, the system would infer that the title of the book is "Animal Farm" and every book has an author. The author for the book "Animal Farm" is "George Orwell". These relations are all stored inside the database and can be used by other modules to discover further information.

5.2.4   Database

The database is the central repository where all the information gathered is stored. A database is typically made up of a set of tables, which define the structure of the data and relationships between the data. Inside the tables, one finds the actual data instances which were discovered by combining the strategies and the Oracles mentioned before. The relationships are discovered when the data is integrated together. The information gathered from each source consists of a group of data which is semantically typed (therefore being an instance of one of the concepts found in one of the ontologies). The data found in this group, although obtained from a source without a formal structure can be easily structured by using the relational information found in one of the ontologies. Eg: if we find a book author in the same context as a book title and we have an ontology containing rules related to books, we would probably find a rule which states that an author writes a book and a book has a title. The problem here is how to find a good context. The simplest way to solve this problem is to use proximity. If the name of a person appears somewhere in the text not far

from the title of a book, then the person is considered a potential author. Obviously, this is not good enough for our needs since this heuristic does not guarantee that the person is actually the author of the book, but because we have several Oracles all validating the same data, then we can check if the person is the actual author of the book or not (Eg: in this case, we could have an Oracle that checks authors and book titles using the Amazon web service). This example is not only relevant for data found in the same context, the same approach is also used for data found in different contexts or even documents. Eg: Imagine that in a particular page we find the name of a person in the proximity of a book title and in a different page we find the book title in the proximity of an ISBN number. A search in one of the ontologies reveals that a book has an author and an ISBN number. Using this rule, the three items of data are integrated together. In this case, the integrating element was the book title since it was common in both contexts. Once again, the validity of the data and their relationships is checked by the various oracles. There is a tight coupling between the ontologies and the database because the information extracted from the documents are just instances of the concepts found in the ontologies. Therefore it makes much more sense to store the ontological relationships inside such a database as well.

5.2.5 The Armadillo methodology

The Armadillo methodology annotates large repositories with domain-specific annotations. It performs this task by harvesting and extracting information from different sources which is then integrated into a common repository. The methodology is largely unsupervised meaning that the user's intervention is very limited.

The approach adopted in this methodology is the following; the user provides:

1. an Ontology (or a set of Ontologies) which define the domain being processed.

2. a set of possible instances (each of which is associated with a concept in the ontology).

3. a reference to a set of documents that need to be annotated (can vary from a small set of documents to the whole Internet).

The set of possible instances (referred to previously as strategies) can be expressed using either a list of instances or using generic patterns capable of spotting a whole set of potential instances. There is no need for the instances discovered to be highly reliable since they will be confirmed at a later stage using other strategies (E.g. multiple occurrences in other documents, etc.).

The cycle begins by going through a subset of documents available for processing and discovering possible instances using the partial set of instances already available. When new instances are discovered, they are;

1. confirmed by the Oracles by using a number of methodologies such as using multiple evidence from different sources (E.g. a new piece of information is confirmed if it is found in different linguistic or semantic contexts).

2. if they are valid instances they are;

   (a) integrated with the existing data found in the central database (E.g. some entities are merged, updated, etc.).

   (b) used by the Strategies to discover new seeds.

   (c) used by the Oracles to validate new data.

3. otherwise, they are discarded.

This cycle continues until there is no more information to discover. The result of this whole process is a database containing structured data which can be subsequently used for several tasks (such as semantically annotating documents, etc).

The Armadillo methodology can be applied to any domain. In fact, to port it to new applications does not require knowledge of IE. All the methods used are domain independent and based on generic strategies. The only domain specific parts are the initial settings. The next section will take a look at how the implementation of the methodology works.

5.2.6 The methodology at work

The internal workings of the Armadillo methodology (See Figure 5.2) are quite straightforward too. To make use of the system, one has to go through two phases, the first phase is the domain setup aimed at the Domain Architects and the second one is the actual system usage aimed at the normal user.

Domain Setup

In the Domain Setup phase, a Domain Architect is required to plug together the different strategies, Oracles and ontologies. The database is hidden from the user since it is part of the ontology handling mechanism. Whenever a new instance is inserted in the ontology, the system automatically inserts it in the underlying database. The insertion process is transparent and does not impose any additional overhead on the system or on the user. The Domain Architect must perform two main tasks, the first task is to define the various modules and the second task is to attach them together.
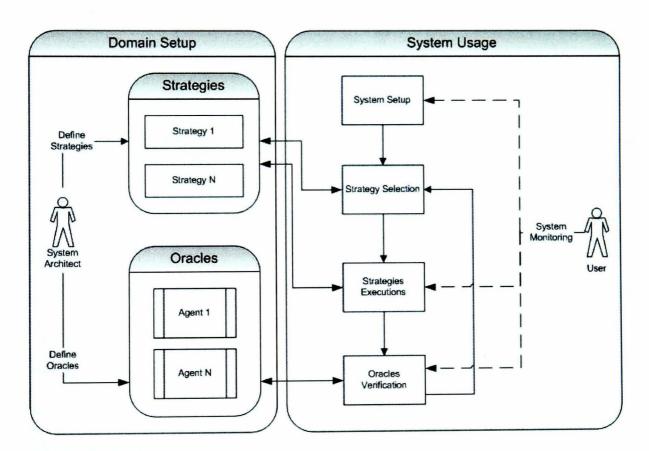
Figure 5.2: The nuts and bolts of the methodology.

To facilitate this task, Armadillo provides a Domain Setup interface designed to ease the life of the Domain Architect (See Figure 5.3). In this interface, the user is not expected to do any programming (although it does not limit the user from doing so). The first steps in creating a domain include:

Loading an Ontology A user can add one or more ontologies to the system. The ontologies are used to type all the data in the system and glue the different modules together. The interface loads ontologies written in DAML+OIL.

Loading a Reusable Component A reusable software component (such as a Java Bean) is a self-contained object that can be controlled dynamically and assembled to form applications. Component software environments, along with object-oriented techniques, allow applications to be built using a building-block approach. For such an approach to work, components must interoperate according to a set of rules and guidelines, and must behave in ways that are expected. The system provides a common interface capable of discovering the several methods found inside the component. It also allows the user to simply add new components. The inputs and outputs of these methods are then typed using the ontologies loaded in the system. If the user does not find adequate components to include in the system, they can always be created programmatically.

Creating or Loading a Gazetteer A user can load existing lists and gazetteers to the system. When a gazetteer is loaded, its contents can be modified using the interface. Once again, the elements found in these lists must be typed to a concept in the ontology. E.g. a gazetteer containing the elements "China, France, Italy, United Kingdom, etc." is typed with the concept in the ontology

called "World Countries".

Creating a Heuristic A heuristic is similar to a gazetteer in the sense that it represents a class of objects. The main difference is that whilst a gazetteer lists the instances belonging to that class, the heuristic allows a user to define a rule using regular expressions[6] to define all those instances. As an example, lets take a look at a heuristic that discovers names of people. Using regular expressions we could easily write a pattern that retrieves all the bigrams whose first letter is a capital letter.

Binding to a Web Service Web services are software components containing a set of functions that can be accessed remotely using TCP/IP as the transportation medium. They are quite similar to the Java Beans we saw earlier. The main difference is that they are accessed remotely and run on another machine somewhere over the internet while a Java Bean is executed locally. The system provides a common interface capable of discovering the several methods found inside the web services. The inputs and outputs of these methods are then typed using the ontologies loaded in the system. If the user does not find adequate web services to include in the system, they can always be created using standard web service techniques in any computer language available.

Creating a Wrapper A wrapper is a template of the data and layout surrounding information which needs to be extracted in a set of documents. It is normally used by software agents who need to locate and extract data from a set of similar documents. Lets take as an example a corpus of documents made up of seminar

---

[6]A regular expression is a formula for matching strings that follow some pattern.

announcements. In these documents it is quite common to find a starting time for a seminar in the following format "Time: 12:30". The main difference between documents is that the time changes depending on the seminar. A typical wrapper would use the word "Time: " as the data surrounding the time which needs to be extracted and anything that follows it as being the actual information that needs to be extracted.

The system presents to the user a very simple interface to create wrappers, by using positive examples provided by the user himself. The user is asked to mark a couple of examples in the documents and the system learns how to extract these instances. The learning algorithm used is based on simple wrapping techniques and manages to achieve very good results since the web pages are very regular.

Downloading Web Pages This download module was inserted in the system for convenience's sake. Since the basic raw material of the system is a collection of web pages, it makes sense to provide a module that facilitates the task of retrieving them.

Once the creation of these modules is completed, they are stored inside the application for later use. In fact, these modules are not bound to a specific domain and can be reused over and over again even in different applications. Since standard techniques are being used, they can also be easily exported and used in different systems. Although we now have all the tools necessary to create the strategies and Oracles, there exist no connection what so ever between these modules. This takes us to the second task of the Domain Architect.
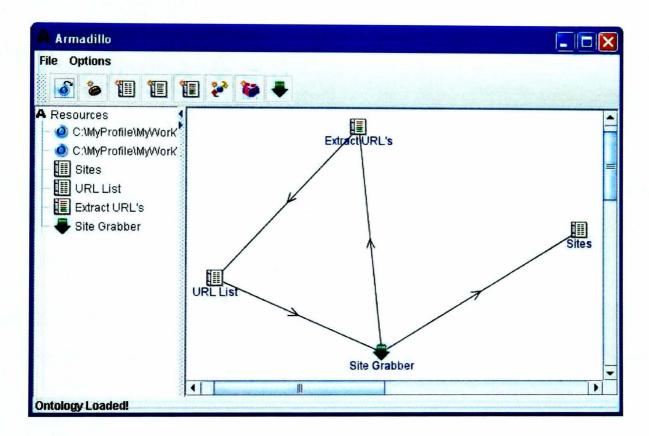
Figure 5.3: The Armadillo Domain Setup.

As can be seen in Figure 5.3, the user must select the components which were created previously, create instances of them and compose them together. In the example shown in Figure 5.3, the diagram in the right panel represent a crawler created in this way. It is made up of four components:

URL List is a module that takes as input a URL or a list of them and return as output the first URL in the list when a request is made.

Site Grabber is a module that consumes a URL and returns as output an object which represents a Web Page (Basically it downloads the web page specified in the URL and returns it).

Extract URL's is a module that given a web page as input, looks for a pattern inside the page that represents a URL. The URL is then extracted and returned as output. The pattern is expressed using regular expressions.

Sites is a module that takes as input a web page and adds it to a list. When a request is made, it returns as output the first web page in the list. It works in a similar way as the URL List but the type of the data is different.

There are a number of things to notice at this stage. First of all, the inputs and outputs are typed using the concepts found in the ontologies. The concepts used are a URL and a Web Page. Secondly, the composition was performed by simply selecting a module and dragging a line to the connecting module. Since all the inputs and outputs are typed, the system can check for semantic type mismatch. Finally, there does not seem to be any entry point for the crawler. The reason is that all modules work using a data driven approach. Therefore, to start operating the crawler, it is

simply a matter of inserting a URL in the URL List module. This can be inserted either programmatically or manually. Immediately, the Site Grabber is called using the newly inserted URL. The module downloads the web page and sends a copy to the Extract URL's module and to the Sites module. The Extract URL's module applies the pattern to extract URLs over the newly downloaded site and sends the resulting URLs to the URL List. The Site module is just a store which can be used for further processing. The cycle continues like this until either there are no more sites to download or until the user decides to stop the crawler.

Although this was a very simple example, there is nothing stopping the Domain Architect from creating much more complex modules. Whenever a module is created, it can be stored as a bean or executed as a web service and used by other modules. As can be seen, the Domain Architect can model the whole domain using these basic yet powerful tools.

System Usage

Since most of the work is performed by the Domain Architect, the role of the user is quite simple. It only involves setting up the initial system and monitoring its execution. The system generally starts from a set of generic strategies defining where to search for information and what to look for. If some initial data is required, it is supplied by the user. Since the whole system is data driven, in order to use the system for a different application of the same domain, it is just a matter of altering the initial settings. In the example of the crawler found in Section 5.2.6, to crawl web site X rather than web site Y, it just involves supplying web site X as the start URL.

When data is harvested using the strategies defined in the domain, it is passed to an Oracle to verify whether the information is valid or not. The task of the Oracle

is to identify which items returned by the strategies are instances of concepts found in one of the preloaded ontologies. Strategies and Oracles are created in exactly the same way using the Domain Setup tool. The major difference is the validity rating which they assign to the data they discover. An item of data verified by an Oracle has a higher rating than one verified by a strategy. Once the data is verified it is stored inside the main database. The system then checks which strategies can be used to process the new information available by matching the type of the new data with the type of the inputs which the modules require. When these modules are identified, they are then executed using the newly added information. This is done to try to discover further information. The results are then given to other more specialised Oracles and the system keeps on looping until either the system manages to discover all the information which can be discovered or the user decides to interrupt the cycle. At any stage, the user can interrupt the execution and inspect the data. He can make any changes required to the data and then resume the system from where it was interrupted. For example if a person name is missed by the system, it can be manually added by the user. The modules that receive as input the output of that name finder will then be re-run and further information will hopefully be retrieved. The data modified has a very high ranking since an Oracle of type human is considered the most reliable Oracle in the system. If one or more modules are not producing good results, the user can reduce the ratings of that module (even if it is an Oracle). This has the effect of reducing the whole impact which that module has on the whole system.

The defined architecture works as a "Glass Box". All the steps performed by the system are shown to the user together with their input and output. The user can check

the intermediate results and manually modify their output, or change their strategy (if possible, such as in the case of modules who integrate information). In this way the user is able both to check the results of each step and to improve the results of the system by manually providing some contributions (additions, corrections, deletions).

Rationale

It is clear that we do not make any assumptions about any particular domain. The strategies range from simple/generic ones (like looking for capitalised words) to more complex ones (like setting up and using IE tools) and they can be easily created for any domain. This technique works because a key feature of the web is the *Redundancy* of information. Redundancy is given by the presence of multiple citations of the same information in different contexts and in different superficial formats. This factor is currently used for improving question answering systems (39). When known information is present in different sources, it is possible to use its multiple occurrences to bootstrap recognizers that, when generalized, will retrieve other pieces of information, producing in turn more (generic) recognizers (15). Information can be present in different formats on the Web: in documents, in repositories (e.g. databases or digital libraries), via agents able to integrate different information sources, etc. From them or their output, it is possible to extract information with different reliability. Systems such as databases generally contain structured data and can be queried using an API. In case the API is not available (e.g. the database has a web front end and the output is textual), wrappers can be induced to extract such information. Wrapper Induction methodologies are able to model rigidly structured Web pages such as those produced by databases (79) (89). When the information is contained in textual documents, extracting information requires more sophisticated methodologies. Wrapper induction

systems have been extended to cope with less rigidly structured pages (47), free texts and even a mixture of them (23). There is an obvious increasing degree of complexity in the extraction task mentioned above. The more difficult the task is, the less reliable generally the extracted information is. For example wrapper induction systems generally reach 100% on rigidly structured documents, while IE systems reach some 70% on free texts. Also, the more the complexity increases, the more the amount of data needed for training grows: wrappers can be trained with a handful of examples whereas full IE systems may require millions of words (85). This is just an example of the idea. The more the task becomes complex, the more information is needed for training, the more reliable input data becomes difficult to identify.

To make the system scalable we have implemented an architecture based on Web Services where each task is divided in subtasks. Each subtask is performed by a server which in turn will use other servers for implementing parts of the subtask. Each server exposes a declaration of input and output, plus a set of working parameters. Servers are reusable in different contexts and applications. For example one server in the CS department task mentioned in Section 5.3, will return all papers written by a person by accessing Citeseer. Another one will do the same on another digital library . The named entity recogniser server (whose role is to decide if a string is a name) will invoke these servers and integrate the evidence returned and decide if such evidence is enough to conclude that the candidate string represents a person.

Facilities for defining wrappers are provided in our architecture by Amilcare (nlp.shef.ac.uk/amilcare/), an adaptive IE system based on a wrapper induction methodology able to cope with a whole range of documents from rigidly structured documents to free texts (25). Amilcare can be trained to work on rigid documents

(e.g. Citeseer or Google output) by providing a handful of manually annotated examples, while it needs some hundreds of examples for more sophisticated cases (24). All the servers are defined in a resource pool and can be inserted in a user-defined architecture to perform some specific tasks. New servers can be easily defined and added to the pool by wrapping them in a standard format. In the CS website task, wrappers are defined for all the resources described in Section 5.3.

5.2.7 Future Development

Although the current implementation of the Armadillo system is based around Web Services, it still executes on a stand alone computer. This was important for the scope of this thesis since when we were testing the core methodology, it facilitated the debugging of the system per se. Now that we are confident that the methodology works fine and produces effective results, we will take a peek at the direction which future enhancements of the system should take.

Due to the distributed nature of the web and with the intuition that a distributed application can exploit better the resources on the web, we decided to model our system on the same concept. In fact, Armadillo was originally designed using a modular architecture. The rational behind our architectural choice was that eventually, we wanted to get each module, separate it from the core architecture and transform it into a self contained entity over the web with as little effort as possible. In doing so, we were not thinking of creating any new standard but rather, make use of the existing ones (such as web services, etc). Also, we did not want to restrict anyone with proprietary formats, but we wanted to make the system easily adaptable by everyone. It is our conviction that if we want the SW to become a reality, a global voluntary effort is necessary whereby everyone contributes a little towards making

the SW a success.

The way forward for Armadillo is basically to repackage the modules either as web services or as mobile intelligent agents. The web service approach has the advantage of exploiting resources which are located somewhere remotely. As such, a powerful central server is not required since the only task of the core Armadillo system would be to coordinate the distributed modules. The problem with web services is that if the server or the connection to the server which hosts an important web service is not reachable, the processing of the Armadillo system can reach a halt until the web service is available again. With mobile intelligent agents, this problem can be partially solved. If we imagine a scenario, whereby a server hosting an agent is going off-line, the mobile agent can easily transfer itself to another server which will stay on-line. Thus, Armadillo will still be able to access the service offered by the agent even if its original hosting server is off-line. A more intelligent way of solving such a problem would be to have similar mobile agents, located on different servers collaborating together on the same task. Thus if one of them goes off-line, it is just a matter of distributing the work, assigned to the off-line agent, to the other similar agents that are still online.

Another future enhancement of the system relates to the composition of services. The original system allowed the users to define a list of services which are executed in a sequential order. The final version of the system presented in this thesis, went a step further and offered a graphical user interface. This gave the users, the facility to plug-in different components visually and create a flow diagram whereby several components can be executed in parallel. Unfortunately, this system is still too complex to configure for non-expert users. This complexity arises from two essential tasks,

first of all, the user must find and define the services to use. Secondly, the user must also compose the services manually.

The discovery of services can be automated to a certain extent. If we make use of systems like the Universal Description, Discovery and Integration[7] (UDDI) directory we could make it possible for users to discover available web services without going into too much hassle.

The composition of services is still a complex task which is very hard to automate. There is currently heavy research going on, regarding ways in which this can be done by using planners. These programs have special purpose algorithms which take as input some problem and goals in a planning domain, and returning a plan to meet those goals, or a determination that no plan is possible. If we add a learning algorithm to the planner and use a methodology similar to the one developed in Chapter 4 (whereby a learner is being trained by the user), we might be able to automate simple repetitive tasks and eventually move towards automating more complex tasks.

In the original system, we also made us of the recognise-act cycle whereby whenever data is harvested ...

1. The data is identified as being an instance of a concept found in one of the ontologies.

2. A search is made through all the services available in order to recognise which services accept the newly discovered data as input.

3. The system acts upon this new found data by invoking the service using this data.

---

[7]A directory model for web services. UDDI is a specification for maintaining standardised directories of information about web services, recording their capabilities, location and requirements in a universally recognised format.

4. The cycle continues until there is no more data which can be consumed by the services.

Although this system worked quite successfully, it is our believe that in future systems, it should be researched and exploited further. Such a producer-consumer approach can help reduce a lot the need of composition since the composition of services can be induced from the type of the available data.

5.3 The Computer Science Department Task: A case study

To highlight the internal workings of the methodology mentioned above, we will take a look at the Computer Science Department Task. The goal is to discover the people who work in a specific department and their publication list. For the first task, we are not just limited to the name of the person but also to other personal information such as position, home page, email address, telephone number, etc if this is available. For the second task, the list of published papers must be larger than the one provided by online services such as Citeseer. On the web, we cannot really assume anything about the structure and layout of the web pages. They will probably be different for each and every department but we can expect to find some higher level constructs, such as a page listing all the people working in the department and many other constructs. These are all cues which the Domain Architect inserted inside the system when it was being designed.

Finding People Names

The task of finding people's name who work in a specific department is the first step towards starting the whole process. Without these names, one cannot find the

related details and publications. At first, it might seem like a trivial task but let us not mix it with the same problem which a generic Named Entity Recognition caters for. This task is much more complex because the site contains many irrelevant people's names, e.g. names of undergraduate students, clerks, secretaries, etc, as well as names of researchers from other departments that e.g. gave seminars in the department, participate in common projects or have co-authored papers with members of staff. One could still make use of a generic Named Entity Recogniser (NER) such as Annie (www.gate.ac.uk) but this option is not always viable. Apart from recognising ALL the people's names in the site (without discriminating between relevant and irrelevant), if we apply it on a normal web site the classic NER would be very slow. We performed some initial experiments using this approach on the CS department site at the University of Southampton. This site was chosen because it was neither too big nor too small and contains around 1,600 pages most of which are generated automatically by a database (therefore containing a very regular structure). The experiments showed that an NER tends to be quite slow if launched on such sites and can be quite imprecise on Web pages, as they are generally defined for newspaper-like articles. Because of this, we use a short list of highly reliable seed names discovered either by using simple strategies or else we utilise a NER in a very restricted way (i.e. rather than applying it on a whole document, we use it on a subset of the document). These names are then used as seeds to bootstrap learning in order to find further names.

Finding Seed Names

The search for seed names makes use of a number of weak strategies combined with Oracles. The process works as follows:

1. The system crawls the web site looking for strings that are potential names of people (e.g. using a gazetteer of first names and a regular expression such as **<first-name> followed by a capitalised word**, using a NER, etc.).

2. As soon as the system has some potential names, it makes use of the following Oracles to try to validate those names:

   - **AKT 3store**

     (inanna.ecs.soton.ac.uk/3store/):

     – Input: the potential name;

     – Output: a list of papers, co-authors, personal details and a URL for the home page (if any);

   - **Annie**

     (www.gate.ac.uk):

     – Input: the potential name and the text surrounding it;

     – Output: True/False indicating whether the string is a name or not;

   - **Citeseer**

     (www.citeseer.com):

     – Input: the potential name;

     – Output: a list of papers, co-authors and a URL for the home page (if any);

   - **Google**

     (www.google.co.uk)

     – Input: the potential name and the URL of the site in order to restrict search;

— Output: Relevant Pages that are hopefully home pages;

- **HomePageSearch**

  (hpsearch.uni-trier.de):

  — Input: the potential name;

  — Output: a URL for the home page (if any);

- **The CS bibliography at Unitrier**

  (www.informatik.uni-trier.de/ ley/db/):

  — Input: the potential name:

  — Output: a list of papers and co-authors (if any);

The information returned by the digital libraries (AKT 3store, Citeseer and Unitrier) is used to confirm or deny the name identity of the string. If the results they return are reasonable for a specific name, this name is retained as being a potential name. At this stage, it is important to understand what we mean by a reasonable result and why we included this restriction. A reasonable result is one where the system returns neither too many hits nor too few. If a string is a valid name, we expect to have a number of papers returned, otherwise the output is either empty or with unlikely features.

For example, when querying these digital libraries with a popular term like "Smith", AKT 3store, Citeseer and Unitrier return more than 10,000 related papers each. This is definitely not a reasonable result since the probability that a person writes more than 150 papers is quite low. In these cases, the name is discarded. The opposite happens when looking for a non-name (e.g. the words "Sea Lion"), no papers are returned and therefore the potential name is not

accepted. The criterion we use is quite restrictive and generally accepts a name when the system returns more than 5 papers and less than 50. These upper and lower bounds were decided empirically and were based upon which thresholds keep reliability high. The lower bound may seem small but as already noted by Brin (15), the redundancy of information allows to bootstrap learning using just a limited amount of information. The results of the digital libraries are integrated with those of the classic Named Entity Recogniser run on a window of words around the candidate (so to avoid the problem of slow processing). The result of this process is a set of people which fall into three main categories:

(a) those working for the department;

(b) those who do not work there, but they are cited because, for example, they have collaborated with some of the researchers of the department;

(c) and false positives (those who are not even names of people).

Out of the three categories, only the people in the first one are correct. The remaining people must be filtered out. For this reason, other Oracles are used such as AKT 3store, Citeseer, Google and HomepageSearch to look for a personal web page in the site. If such a page is not found, the names are discarded since in general, most people have their personal web site. The personal web pages are recognised using simple heuristics such as looking for the name of the person or the words "Home Page" in the title.

The different strategies mentioned above are all weak, and they all report high recall but low precision. This does not matter much because they were only meant to determine a small reliable list of names, which can be used as seed

to bootstrap the learning phase. Since each weak process is not working in isolation but rather in collaboration with the others (by using results obtained from the other modules), their combination manages to produce highly accurate data.

Learning Further Names

Starting from a list of seed names makes the discovery of further names easier. These seed names although distinct, have one common feature, they all represent people working in the same place. The strategies we used previously therefore gave us a subset of people working in a particular department. If we manage to find other pages where the people in this subset are mentioned together, we might be able to discover new names of people that belong to this set using more refined strategies. To do this, we make use of Google to try to find pages where these people are mentioned together.

Once we have these pages, all the occurrences of seed names are then annotated on the retrieved documents. Initially, the program looks for HTML structures such as lists and tables containing names which were annotated. Such structures normally have an implicit semantic relationship. Lists generally contain elements of the same type (e.g. names of people), while the semantics in tables is generally related to the position either in rows or columns (e.g. all the elements of the first column are people, the second column represents addresses, etc.). If a reasonable quantity of known names (at least four or five in our case) is found, learning is performed on these sub parts of the documents. A classifier capable of learning linguistic and/or formatting features (e.g. relevant names are always the first element in each row) is trained and used to identify where the names appear. If we succeed, we are able to

reliably recognise the other names in the structure. [8].

In the approach mentioned in this section, we are assuming that a department contains lists and tables of some kind (including a staff list). This is a valid assumption which we obtained from our experience after many years spent browsing through web sites. Our search is not only limited to simple lists or tables, but other more complex structures can be useful as well provided we can discriminate between the information found in these structures. One typical example commonly found in these web sites is a table assigning supervisors to students. In this case a strategy could be used which discriminates between supervisors and students by retaining only the names found in a particular column of the table.

When new examples are identified, the site is further annotated and more patterns can potentially be learnt. New names are cross-checked on the resources used to identify the seed list such as the digital libraries. In our experiments this is enough to discover a large part of the staff of an average CS website with very limited noise, even using a very strict strategy of multiple cross-evidence. To accept a learnt name, we use a combination of the following strategies:

1. either the name was recognised as a seed by the previous strategy;

2. or the name is included in an HTML structure where other known occurrences are found such as a list;

3. or there is evidence from generic patterns (as derived by recognising people on other sites) that this is a person (inspired by (87)).

---

[8]Classifiers are induced in our implementation by Amilcare (http://nlp.shef.ac.uk/amilcare/), an adaptive IE system developed at Sheffield (25)

Extracting Personal Data

Personal data such as email address, telephone number, position, etc is slightly more tricky to find. This information is normally found in one place, generally inside the personal home page (or a page with a similar role) of that particular person. Once again we combine information from Citeseer, HomepageSearch and Google to check if the person has a known page in the current web site. Otherwise we look for occurrences in the site in which the name is completely included in an hyperlink pointing internally to the site. We must be careful where the links lead to. In many departments, these links lead to intermediate pages automatically generated by a database. Heuristics can be used to help us avoid such pages.

When we manage to reach a personal page, we make use of a named entity recogniser (e.g. Annie) to easily identify the personal data. This analysis is not just performed on the main page but also on subpages pointed to by the main home page. It is quite common in a home page to have contact details mentioned in a sub page inside the web site. To avoid falling into the trap of following links all over the web, we consider only pages with an address under the same domain.

Discovering Papers

Discovering the title of papers and their authors is a very difficult task. First of all there is no standard format in which publications are listed. If the publication pages are formatted in the same way, it would be easy to extract the information but this happens very rarely. In most cases each person writes the list using a personal format. Even in these cases, it is very rare that the person uses the same style in the entire page and very often the style is quite irregular since the list is compiled manually in

different times. This is a typical case in which the classic methodology of manually annotating some examples for each page for each member of staff is infeasible, due to the large number of different pages. Also irregularities in style produce noisy data and classic wrappers are not able to cope with noise. A generic methodology is needed that does not require any manual annotation.

Secondly, although there is no written rule, normally a publication list contains a list of papers having the title of the paper, the conference where it was published and the authors of the paper. A title is normally a random sequence of words (e.g. the title of (39)) and cannot be characterised in any way (i.e. we cannot write generic patterns for identifying candidate strings as we did for people). A title can also be very similar to the name of a conference and it might be hard to distinguish which is the title of the paper and which is the name of the conference where it was published. Finding out the names of the authors is not difficult as we have seen in the previous section. The problems arise when we try to associate authors with a particular paper. There is nothing apart from visual cues which create the association. Unfortunately, we cannot characterise these cues since there are no rules which state whether the name of an author must appear before or after the title of a paper. Authors are names in particular positions and in particular contexts. They must not be confused with editors of collections in which the paper can be published, nor they must be confused with other names mentioned in the surrounding text.

Finally, most publications have more than one author, it is very possible that each paper is cited more than one time even in the same web site by different co-authors. Nearly every department and member of staff in a CS department provide a personal updated list of publications.

The strategy we propose uses the names of staff members as keywords to query the digital libraries (AKT 3store, Citeseer and UniTrier). These digital stores return a list of papers for every staff member. Since all the available libraries are based upon manual or semiautomatic techniques to harvest the information, all the lists retrieved are incomplete. This is not a problem because we only require a small number of examples to bootstrap the paper discovery process. The next step is to use the titles in the list to query a search engine and get a number of pages containing multiple paper citations (We accept only pages which contain at least four paper titles). Titles are used because they tend to be unique identifiers. Pages with few hits are discarded since it is unlikely that they contain other examples for the same author. Pages with many hits are discarded too so as to avoid titles which contain very common strings such as "System".

Once we have a set of pages containing good examples, we take each page and annotate the titles of the papers and the names of the authors. Examples stored in HTML lists and tables for which we have multiple evidence, are favoured over other examples since it is much easier to divide the page into smaller chunks and extract those chunks. Each chunk will contain a complete set of data elements, consisting of one paper title together with the related conference name (where it was published) and its authors. Please note however that the structure of the citation is often not very structured internally. The following two examples refer to chunks representing the same publication but they come from two different web pages[9][10].

Yorick Wilks: On-line Curriculum Vitae

```
<p>
<span lang=FR style='mso-ansi-language:FR'>
```

---

[9]http://www.dcs.shef.ac.uk/~yorick/cv.html

[10]http://www.dcs.shef.ac.uk/~fabio/cira-papers.html

```
Dingli, A., Ciravegna, F. and Wilks, Y. (2003)
Automatic Semantic Annotation using Unsupervised
Information Extraction and Integration.
In Skuppin (ed.) Proc. Second Internat.
Conference on Knowledge Capture (KCAP03).
</span>
</p>
```

Publication List of Fabio Ciravegna

```
<li class=MsoNormal style='mso-list:l16 level1 lfo23'>
<span lang=EN-US style='mso-ansi-language:EN-US'>
    Alexiei Dingli, Fabio Ciravegna, and Yorick Wilks
<br>
<a href="paperi\KCAP03.pdf">
    Automatic Semantic Annotation using
    Unsupervised Information Extraction and Integration
</a>
<br>
    in Proceedings of the
</span>
<span class=small>
<a href="http://sern.ucalgary.ca/ksi/K-CAP/K-CAP2003/">
    K-CAP 2003
</a>
<a href="http://km.aifb.uni-karlsruhe.de/ws/semannot2003/">
    Workshop on Knowledge Markup and Semantic Annotation,
</a>
    ,Sanibel, Florida, October 26, 2003
</span>
<br style='mso-special-character:line-break'>
<![if !supportLineBreakNewLine]>
<br style='mso-special-character:line-break'>
<![endif]>
</li>
```

The example clearly shows that simple wrappers relying on the HTML structure only would be ineffective. The number of tags varying between documents varies

according to the different styles and layouts used. HTML is a language that provides only information about the layout and there is no way to discriminate between authors and editors and title of paper and title of collection since there is no semantic information stored inside the document. More sophisticated wrapper induction systems are needed, like that provided by Amilcare (25), which exploits both XML structures and (para-)linguistic information (24).

By adopting a strategy like we mentioned before made up of a cycle of annotation/learning/annotation, we are able to discover a large number of new papers. The cycle terminates when there is no more information to discover or the user decides to interrupt the cycle. Every time co-authorship amongst people is discovered, we exploit the redundancy again by using the other authors to discover more collaborations between the co-authors.

5.4 Armadillo Evaluation

The architecture mentioned is fully implemented and we made extensive experiments on a number of Computer Science (CS) web sites. We have experimented on the sites of (1) the Computer Science Department of the University of Sheffield[11], UK, (2) the School of Informatics at the University of Edinburgh[12], UK, and (3) the Department of Computer Science of the University of Aberdeen[13], UK. In the following sections we extensively report about the results obtained on these sites.

5.4.1 Finding People's Names

The first task was to identify the names of people working in a particular site. It was not enough to return just the names of all the people mentioned in the site since such a list could also include names of students, names of people from other departments who collaborated with the current CS department, etc. Therefore it is not just a matter of discovering named entities but it is a much more complex task. The experiments show that the names of people can be found with a high reliability. The following are the results for the various sites:

The Computer Science Department of the University of Sheffield

In the case of the Sheffield's department using just Information Integration (II) [14] techniques the system discovers 40 seed names of people belonging to the department as either academics, researchers or PhD students, 35 correct and 5 wrong. A qualitative evaluation of the errors is worth doing. The false positives can be divided into

---

[11]http://www.dcs.shef.ac.uk
[12]http://www.inf.ed.ac.uk
[13]http://www.csd.abdn.ac.uk
[14]By II techniques we mean that no IE is involved. Information is just harvested from online sources and integrated together.

three groups:

Spurious Hits refer to entities which are truly wrong but for which a home page was also erroneously found. In general, their number is quite low in proportion to the total number of names. Recognising these names as false hits is quite easy for a person, so if the results are to be used or checked by a person, they do not constitute a problem. The only spurious hit we got (Computer Science) is actually the name of the departmental home page. A tighter post processing filtering technique was also applied whereby potential names containing stop words (such as An, For, The and Is) are discarded. This removed spurious names including the entities "An Journal", "For Information" and "The Is" which were returned by the system in the initial phases, thus leaving only the one reported here in the final results.

Wrong Persons refer to names of people who are researchers in other institutions but which also have a web page similar to a home page in the Sheffield Computer Science department. The list of people include:

- Eugenio Moggi

- Jose Principe

- Sophia Ananiadou

The recognition of these people is due to the wrong identification of some web pages within the departmental site as their home pages. This occurred because in the past, these people all collaborated in some project with someone from within the department and their name is mentioned in a heading of one of the pages in the web site of the department.

Wrong Criterion refers to wrong information found within the web site. This included the name of the following person:

- Florentine Ipate

The system accepts a person as being part of a department if he/she has a home page in the departmental web site. In fact this person does have a home page at the department and further examination reveals that he was in fact a past student or employee. Therefore, the system could not possibly know that he is no longer employed within the department since the old web pages are still accessible and there is absolutely nothing on them that indicates that the particular person no longer studies or work over there.

These potential names are then used to seed the learning algorithm. But before doing so, it must find out the best pages from which to learn. Pages are considered good if they contain lists of people. A search is conducted in the Computer Science web site using a number of queries similar to this:

$$\textbf{\textit{site:dcs.shef.ac.uk}} \; "academics \mid personnel \mid academics \mid staff \mid faces \mid list"$$

$$\textbf{\textit{Lapata Clayton Demetriou}}$$

The first element in the query is the name of the site we are searching through in this case *dcs.shef.ac.uk*. The second element is an *OR* list containing relevant keywords such as academics, personnel, etc. This list is generated automatically from a domain dependent list of words. The final element is a list of seed names and surnames obtained from the previous procedures. These lists normally contain about three names or surnames chosen randomly. In this case the surname Lapata

and Demetriou, and the name Clayton. All of these are people who work in the department and who were discovered earlier. We use just three names because of limitations imposed on us by the search engine. In Google, a query cannot be bigger than a certain length. The system performs around 30 such queries and then selects from the results the web pages that appear the most. The top five web pages obtained from these experiments were:

1. http://www.dcs.shef.ac.uk/department/people/groups.htm

2. http://www.dcs.shef.ac.uk/department/people/atoz.htm

3. http://www.dcs.shef.ac.uk/research/publications.html

4. http://www.dcs.shef.ac.uk/people/atoz.htm

5. http://www.dcs.shef.ac.uk/people/groups.htm

If we examine this list, we find that all the pages (except for the third page) contain structured lists of names belonging to people that work in the department. Therefore four out of five of these pages are easy to learn by using our learning algorithm. The third page in the list is a very unstructured page containing lists of people together with their publications. It obviously ranked well in the search because it contains a list of all the publications in the department and all the people who published papers are mentioned in this page. Since this unstructured page is quite difficult to learn by using a learning algorithm, we expect to have some errors introduced in the Information Extraction (IE) phase.

The Amilcare IE tool we used, managed to discover a total of 84 names from this department. Of these 116 were correct and 8 were wrong. An analysis of the wrong

names (introduced in the IE phase and before filtering) shows that four of them were spurious hits similar to the ones we saw earlier while the remaining three were real errors. These included:

- For Translation

- The Of

- In To

- The Research

- Prospective Student

- Reading Reading

- Piek Vossen

As mentioned before, these spurious errors were filtered by making use of a stop word list since most of them contain a stop word in them. This list was reported here because they add a further insight into the problems which Armadillo is facing. Since the phrases in the list do not even make grammatical sense, it seems to suggest that the system is not delimiting adequately the different sections of a document. What is happening is that tags found in the HTML documents are not being treated correctly. More adequate sentence splitting should be investigated when HTML documents are being analysed in order to prevent these errors from the very start.

The remaining error was a name of a real person but who did not work in the department. His name is "Piek Vossen" and he was mistaken for a person working in the department because he appeared in several pages of the CS web site. A search

through the site reveals that in the past he collaborated paper writing with some people working in the department.

While browsing through the Sheffield's CS department web site looking for such anomalies, we came across several other sites with potential problems. There were several people who did not work in the CS department but who had a web page dedicated to them. These consisted mainly of people who gave a seminar in Sheffield. In other cases, pages similar to home pages were found in the home page of one of the members of the department. Apparently, she keeps a bibliography of the publications of these people in separate pages dedicated to them. In our experiments, these people were not considered part of the CS department because probably, their name does not appear in the main lists of people. In other sites, such pages with names could cause problems because even though the model we use to spot a home page of a person works in most cases, there are definitely some exceptions. This means that the heuristics which we use should be refined further in order to cover as much as possible only positive cases.

To evaluate these results, we manually counted the number of academics, research staff and research students found in the groups page[15] (and filtered out secretarial staff which were also included in the list). The total number of people in the department was of 139. If we now calculate Precision, Recall and F-Measure for the data obtained so far, we get the results in Table 5.1.

These experiments show a drastic increase in both precision and recall when Information Extraction (IE) is applied on top of results obtained using Information Integration (II). Precision increases by almost 7%[16] while recall increases by more

---

[15]http://www.dcs.shef.ac.uk/people/groups.htm

[16] $\frac{100\%}{87.5\%} \times 93.5\% = 106.9\%$ obtaining almost a 7% increase

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
II	139	40	35	5	104	87.5%	25.2%	39.1%
IE	139	124	116	8	23	93.5%	83.5%	88.2%

Table 5.1: Results in Discovering People and Associated Home Page. The first line refers to accuracy in discovering names using the procedure for seed names (Cite-seer+Google, etc.), the second one refers to the discovery using adaptive IE.

than 200%[17] over the II results. The results indicate that the II techniques are good to boost up precision while recall is fairly low. The main reason for this is that the searches are directed by the web page (since potential names are harvested by crawling the web page). This means that since the system has no specific entry point to the web site, it is never guaranteed that it will find links that will take him to a page rich in information (in our case rich in names). Therefore the data being collected has very low recall since the few examples collected are purely by chance and using simple heuristics. Also, the II methods are restricted to a limited number of databases or Oracles. Therefore, their knowledge cannot be greater than the knowledge found in these Oracles. This restriction keeps recall quite low since the system is dependent on these sources. Precision on the other hand rockets up because we are collecting information from these Oracles which have very structured sources. Therefore the wrappers we create manage to collect precise information without any errors.

When we use IE, the story changes. Recall rockets up while precision increases too but the increase in precision is minimal. This increase in recall occurs due to the fact that the IE is not just being applied on an arbitrary web page but on a carefully

---

[17] $\frac{100\%}{25.2\%} \times 83.5\% = 331.4\%$ obtaining around 231% increase

selected page rich in names. The page is selected by querying an Information Retrieval engine (in our case Google) using the information discovered in the previous stage as query terms. The retrieved page is then annotated using very precise data obtained from the II stage so when the IE engine trains, it is training on good examples. It then learns from the structure of the particular page how to recognise names and returns new examples. The slight increase in precision is still very good. Normally, it is expected to have a decrease in precision because the IE engine tends to introduce new errors but the quality of the examples used for training, makes it possible for the IE engine to maintain the same precision. Recall is obviously increased since we are now using a more robust classifier capable of finding new instances of the target concept, trained on very good examples and which is applied on carefully selected pages.

The School of Informatics at the University of Edinburgh

In the experiments performed on the site of the School of Informatics at the University of Edinburgh the system using just II discovers 13 seed names of people working in the department, 11 correct and 2 wrong. According to the lists found on their web site[18][19], the University has a total of 216 people working over there made up of academic staff, research staff and fulltime postgraduates.

If we take a look at the type of errors returned back by the system, we find out that they are very similar to the errors found in the Sheffield site. The first error included a spurious phrase like "Computer Science". The second error was a name of a real person called "Javier Esparza". This happens to be the name of person who

---

[18]http://www.inf.ed.ac.uk/research/iccs/directory.html
[19]http://www.inf.ed.ac.uk/research/iccs/students.html

used to work in Edinburgh many years ago. In the site there are several references pointing to this person because he published and co-authored papers with people who still work over there. Since the system managed to find a copy of one of these papers within the CS web site having his name as the title of the page, the system thought that it was a valid personal page and accepted the name of the person.

The next phase makes use of several IE strategies like the ones mentioned earlier. In this phase the results we obtain depend on a number of factors. When we search for a page, the queries are designed to look for lists of names which can be easily learnt by the IE engine. These pages are generally very structured and therefore, few examples are required to bootstrap the learning process. In this phase, we cannot exclude the possibility of analysing a few unstructured pages, therefore the IE system returns some spurious results. In fact, the IE tool managed to discover a total of 163 names from this department. Of these 153 were correct and 10 were wrong. An analysis of the wrong names (introduced in the IE phase) shows that they were all spurious hits similar to the ones we saw earlier. These included:

- Students Home

- Postgraduates Research

- Management Science

- Intranet Home

- Computer Literacy

- Computer Programming

- Cognitive Science

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
II	216	13	11	2	205	84.6%	5.1%	9.6%
IE	216	163	153	10	63	93.9%	70.8%	80.7%

Table 5.2: Results in Discovering People and Associated Home Page at the School of Informatics at the University of Edinburgh. The first line refers to accuracy in discovering names using the procedure for seed names (Citeseer+Google, etc.), the second one to the discovery using adaptive IE.

- Centre Research

To evaluate these results, we manually counted the number of academics, research staff and research students found in the groups page[20] [21] (and filtered out secretarial staff which were also included in the list). The total number of people in the department was of 216. If we now calculate Precision, Recall and F-Measure for the data obtained so far, we get the results in Table 5.2.

These experiments show a significant change in both precision and recall when IE is applied on top of the II results. Precision increases by almost 11%[22] while recall increases by more than 1200%[23] over the II results. Once again, we are confirming what we saw earlier, basically the fact that the II techniques are good to boost up precision and the IE techniques are good at boosting recall. The increase experienced in this experiment shows that only a few quality data items (from 11 correct names to 153 correct names) are required to bootstrap the whole process and obtain excellent results. The combination of both these techniques seems to be the answer towards

---

[20]http://www.inf.ed.ac.uk/research/iccs/directory.html

[21]http://www.inf.ed.ac.uk/research/iccs/students.html

[22] $\frac{100\%}{84.6\%} \times 93.9\% = 111\%$ obtaining almost an 11% increase

[23] $\frac{100\%}{5.1\%} \times 70.8\% = 1388.2\%$ obtaining around 1288% increase

achieving high precision and recall in knowledge extraction applications.

The Department of Computer Science of the University of Aberdeen

The Computer Science department at the University of Aberdeen is quite small when compared with the other departments. It is composed of 70 people made up of academics, research staff and research students. When Armadillo was tested on this site, the results obtained were quite similar to those already seen. Using just II techniques the system discovers 22 seed names of people belonging to the department as either academics, researchers or PhD students, 21 correct and 1 wrong. An analysis of the errors shows that the name belongs to an ex PhD student of the department. In the web site, one can still access a site with the name of this person on it. There was also another error which i did not include in these results. The name found belonged to an ex member of staff but in this case, it is curious to note that the name of the person is actually still mentioned in the staff page[24]. The staff page contains two additional lists entitled "*Honourary Staff*" and "*Former Staff*". The former list is made up of 5 people from other Universities who were given an honourary title by the University of Aberdeen. The latter list contains the names of two people who used to work at the University. For the purpose of this evaluation the people found in these lists were included as part of the total number of people working in the department. Most of them still have a personal web page somewhere in the site and there are several references to them scattered here and there. In order to figure out who still works in the site, we had to manually go through the site, analyse what was written and deduce from the various clues whether or not the person still works at the CS department. This obviously involves a higher level of intelligent inferencing, much

---

[24]http://www.csd.abdn.ac.uk/people/

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
II	70	22	21	1	49	95.5%	30%	45.7%
IE	70	65	63	2	7	96.9%	90%	93.3%

Table 5.3: Results in Discovering People and Associated Home Page at the Department of Computer Science of the University of Aberdeen. The first line refers to accuracy in discovering names using the procedure for seed names (Citeseer+Google, etc.), the second one to the discovery using adaptive IE.

higher than Armadillo can handle. Because of this, if we did not include these people as positive results, the evaluation would be unfair since it would be unreasonable to expect the system to identify them as persons working at the University.

In the next phase, the IE tool managed to discover a total of 65 names from this department. Of these 63 were correct and 2 were wrong. An analysis of the only wrong name (introduced in the IE phase) shows that it was the name of a past M.Sc. student called "Richard Martin". Although his name is not included in the staff page, he does has some publications and also a web page dedicated to him. Therefore, it was difficult for the system to figure out whether or not the student is still there or not since the different cues associated with this student matched cues associated with typical researchers working at the department (i.e. they must have a home page and a number of publications). A possible solution to this problem could be to analyse and keep record of the date when a piece of information was created on the web. In this way, papers which were published or web pages which were updated many years ago should not be considered as valid clues towards deciding whether or not a person still works in a particular site.

To evaluate these results, we manually counted the number of academics, research

staff and research students found in the groups page[25]. Then we calculated Precision, Recall and F-Measure for the data obtained up to that stage and we got the results in Table 5.3.

These experiments continue to confirm the observations we saw in the other experiments. Precision increases by almost 2%[26] while recall increases by 200%[27] over the II results. This seem to strengthen more our hypothesis that a collaboration of II and IE techniques is necessary to obtain results with high precision and recall.

5.4.2 Paper Discovery

After harvesting all the names of the people working in a CS department, the next task is to identify their publications. The task of discovering papers is very complex. If we analyse the implications associated with finding the title of a paper we realise that a title:

- is made up of a variable number of words.

- is generally a grammatical phrase or sentence but there is no rule binding it from being so!

- it can contain any word, number or symbol in any order!

All cues are not very helpful at identifying a name. If we try to examine the context in which paper titles appear, we realise that this is slightly more regular (since it generally contains the names of the authors, conference name, conference date, conference location, etc) but the author is not really bound to include any

---

[25]http://www.csd.abdn.ac.uk/people/

[26] $\frac{100\%}{95.5\%} \times 96.9\% = 101.5\%$ obtaining almost an 2% increase

[27] $\frac{100\%}{30\%} \times 90\% = 300\%$ obtaining an additional 200% increase

information and the format is left to everyone's discretion. Unfortunately, there is not really a standard way of representation. Some formats are more prevailing than others (like BibTex[28]), but we cannot assume the system will encounter just a particular format because on the net there can be an infinite number of different formats. The only thing we know for sure is that references to publications do not just appear anywhere on the web but in specific places. These include databases containing lists of publications, an author's personal publication list, references found in other papers, etc. Once again, most of these pages contain lists of papers in no particular format but they are definitely much regular.

The experiments we performed were based on the ideas presented earlier. While harvesting the names of people working in a particular site, we also gather a partial list of their publications (from the online databases). For every author, we used this list to query a search engine and obtain a web page where all these publications appear. The rationale is that a page which contains the elements of a representative partial list, will probably contain other unknown members of the complete list. This is the same idea which we used to identify the names of people earlier.

The results presented in this section refer to the experiments performed for the people working in the Sheffield CS department only. In our experiments a paper was considered correctly assigned to a person if it was a paper mentioned in the personal papers list of the author and the title is 100% correct. The seed procedure discovers 143 papers, the learning procedure returns 407. If we manage to discover less than 3 papers for a particular author, we do not consider this list as being representative

---

[28]BibTeX is a program and file format designed by Oren Patashnik and Leslie Lamport in 1985 for the LaTeX document preparation system. The format is entirely character based, so it can be used by any program. It is field based and the BibTeX program will ignore unknown fields, so it is expandable. It is probably the most common format for bibliographies on the Internet.

enough and therefore we exclude the author from the search. Checking the correctness of the results is very labour intensive, therefore we randomly checked the papers extracted for 10 of the staff members for which the seed papers exceeded 5 examples. These are shown in Tables 5.4 and 5.5.

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
R1	12	5	5	0	7	100%	42%	59%
R2	45	7	6	1	38	86%	13%	23%
R3	49	22	22	0	27	100%	45%	62%
R4	94	18	18	0	76	100%	19%	32%
R5	48	8	8	0	40	100%	17%	29%
R6	43	5	5	0	38	100%	12%	21%
R7	81	27	27	0	54	100%	33%	50%
R8	21	8	5	3	13	63%	24%	34%
R9	31	18	15	3	13	83%	48%	61%
R10	103	25	24	1	78	96%	23%	38%
Total	527	143	135	8	384	94%	26%	40%

Table 5.4: Seed Paper Discovery Accuracy; each line represents the papers discovered for a person. Possible represents the number of papers present in the personal publication list page, Actual the number of papers returned by the systems. The actual results are divided in Correct, Wrong and Missing.

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
R1	12	8	8	0	4	100%	67%	80%
R2	45	40	36	4	5	90%	80%	85%
R3	49	36	36	0	13	100%	73%	85%
R4	94	82	50	32	12	61%	53%	57%
R5	48	43	41	2	5	95%	85%	90%
R6	43	20	15	5	23	75%	35%	48%
R7	81	42	33	9	39	79%	41%	54%
R8	21	18	17	1	3	94%	81%	87%
R9	31	28	27	1	3	96%	87%	92%
R10	103	90	83	7	13	92%	81%	86%
Total	527	407	346	61	120	85%	66%	74%

Table 5.5: IE-based Paper Discovery Accuracy obtained by Amilcare using the seeds in table 5.4 to bootstrap learning.

	Possible	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
II	527	143	135	8	384	94.41%	25.62%	40.30%
IE	527	407	346	61	120	85.01%	65.65%	74.09%

Table 5.6: Paper Discovery: Grand total

The results follow what was noticed earlier in the other experiments, that the use of IE increases significantly the overall recall rate when seeded with results obtained using II. In fact, the recall grows from 26% (for seeds) to 66% (for IE-based), an increase of more than $\frac{1}{3}$ of the total number of papers. Precision from 94% to 85% and F-measure from 40% to 74% (see Table 5.6).

5.4.3 Other domains

Armadillo works well in the CS domain discussed in Section 5.4.1, but what about other domains? It is useless creating a system so bound to one particular domain that it cannot be used in other domains. In order to test the portability of the system, myself together with a colleague of mine Mr. Sam Chapman decided to port Armadillo to another domain. Subsequently, the evaluation of this system (which follows) was performed exclusively by myself. For these tests we chose a totally different domain, the Artists domain.

This scenario was created for the ArtEquAKT project[29]. The ArtEquAKT group is building a system which accepts as input a name of a famous artist and the system will generate a biography of that artist on the fly. The aim of this project is to use knowledge acquisition and analysis techniques to extract information from web pages

---

[29]http://www.artequakt.ecs.soton.ac.uk/

on a given subject domain and construct a knowledge base overlaid with an ontology. The ontology together with story templates can then be used to construct stories. This is achieved by walking through the ontology, extract appropriate knowledge from the knowledge base underneath and insert them in the templates. The data they need is not obtained from any particular database but it is being harvested from the web on the fly. A major problem they encountered is how to find the information and once it is found, how to extract it. Because Armadillo was designed to perform exactly this task we decided to take over the challenge and adapt our system for their domain.

For the Artist scenario, we customised the basic Armadillo system and produced Art-madillo. This system basically takes as input a name of an Artist and returns a list of paintings for that particular artist together with other meta information such as dates, etc (if they are available). It works in a similar way to the CS domain seen previously. The only difference being that the main sources of information are related to artists rather than academic researchers. The sources used are various, some of which are listed in Table 5.7.

When the system is given the name of an artist, it searches through these sources for information concerning that particular artist. The newly discovered information is all stored in a database within Art-madillo. As soon as the system has some information about an artist which can be used to bootstrap further information, a search is launched using Google to find new sites which might contain further information. These sites are annotated with the known information. IE is then used to learn the patterns in the site and reapplied on the same site in order to discover new facts. The cycle continues like that until no further information can be discovered.

Site	Artists
http://www.artcyclopedia.com	Cezanne, Raphael
http://www.artpublishing.com	Cezanne, Monet
http://www.artrenewal.org	Caravaggio, Manet, Raphael, Renoir
http://www.bildindex.de	Cezanne, Manet
http://www.postersposters.net	Manet, Monet
http://www.postershop.com	Cezanne, Renoir
http://www.wholesaleoilpainting.com	Caravaggio, Raphael

Table 5.7: A list of sites from where information about particular artists was collected during the manual evaluation.

The following is the result of a comprehensive evaluation which we performed on the system. We took a random set of 6 famous artists and we used them to test the system. A problem we had during the evaluation was how to calculate the total number of pieces created by a particular artist. We could not find any site that gives a complete list of paintings probably because the exact amount is unknown. Therefore, we had to compute a manual estimate of the total number of pieces per artist. What we did was to find one or more sites that list most of the pieces done by a particular artist. Then we went through each site and manually counted the number of paintings. Since the paintings in most sites were listed in alphabetical order, it was possible to count the pieces while going through several sites at once. This was not a straightforward task since we also had to ignore duplicate names and duplicate paintings. There were several cases where some of the paintings were displayed more than once in the searches using the same name but in a different language. In other cases, the same painting was shown using the same name but having different dimensions. This mainly occurred because some of the sites we used

were online shops which sell prints of paintings. The complete list of sites we used for the manual evaluation to calculate the total number of paintings per artist can be seen in Table 5.7. Once we have this estimate, the rest was quite straightforward and basically consisted in going through the results and counting the correct and wrong entries. The results we obtained can be seen in Table 5.8.

Artist	Method	Possible	Actual	Correct	Wrong	Precision	Recall	F-Measure
Caravaggio	II	82	50	50	0	100%	61%	75.8%
	IE	82	81	81	0	100%	98.8%	99.4%
Cezanne	II	188	51	51	0	100%	27.1%	42.7%
	IE	188	88	80	8	91%	42.6%	58%
Manet	II	128	38	38	0	100%	29.7%	45.8%
	IE	128	52	52	0	100%	40.6%	57.8%
Monet	II	130	19	19	0	100%	14.6%	25.5%
	IE	130	73	63	10	86.3%	48.5%	62.1%
Raphael	II	162	97	97	0	100%	59.9%	74.9%
	IE	162	145	140	5	96.5%	86.4%	91.2%
Renoir	II	45	19	18	1	94.7%	40%	56.2%
	IE	45	28	27	1	96.4%	60%	74%

Table 5.8: Artist Discovery

If we analyse the results, we realise that the average precision for all the 6 artists during the II phase was 99.1%. This is not something unexpected since our II sources were structured databases containing thousands of paintings. Therefore, it was easy for our wrapping algorithms to get the names of the paintings and the name of the relevant artist from the sites. The few errors introduced in the system were due to a number of factors some of which already mentioned before. The first kind of error encountered in the sites, was due to the fact that the sites contained duplicates. This was not really a problem since duplicates can be filtered easily from any list. The problems arise when the duplicate name is either spelt differently or else in a different language. If for example we take a famous painting of Renoir entitled "**The Blue Vase**", in various sites, we found the following three versions of the name:

1. The Blue Vase

2. Blue Vase

3. Le Vase Bleu

They all refer to the same painting but for our system, its quite hard to realise that. The second name has a missing article when compared with the original name. We could exclude articles but how can we be sure when an article is part of the name or not? And what happens if we exclude articles and then realise that there is another painting entitle just "Blue Vase"? How would we go about distinguishing them? These are all open questions for which we do not really have an answer but they are definitely worth asking. The third name is 100% correct when compared with the original name, but it is written in French. If a system would analyse this, it would be considered entirely different from the original name (unless the system knows some

French!). A possible solution to this could be to add a translation component to the system. In the previous domain, there was no need for such a component but in the Artists domains, since it is a multilingual domain, such a component could come useful. Unfortunately, translation components are far from being perfect and therefore it would not solve the problem entirely. The second kind of error has to do with wrong entries inside the databases from which we gather information. We have already encountered such an error when we were working earlier in the CS domain. We cannot really do much about this kind of error since as default, we consider the entries inside a database as being 99.99% reliable because for us, a database is an Oracle! If the database is not considered reliable, then it is not used for II in the first place. In the case of the Artists domain, the errors in the databases were negligible therefore they were still used as part of the II techniques. With regards to recall, this depends entirely on the popularity of the particular artist. The more popular the artist was, the more the system is likely to gather information during the II phase. Since as we mentioned before, there is no single database containing a comprehensive list of paintings per artist, we had to use several sources. The recall obtained during the II phase represents the total paintings discovered for a particular artist using different sources.

When it comes to the IE phase, the precision value remains stable or increases in half of the tests while decreases slightly in the other half. This slight decrease was due to the fact that the IE always introduces some errors in the whole process. These errors have to do with wrong boundaries when it comes to detecting names of paintings. On the other hand, recall increases drastically in all cases (and on average it doubles).

We should not forget that the results obtained in the II phase indicate the popularity of the artists in the sites we used as a source for our information. Basically, this means that the more represented an artist is in that site, the more we manage to find structured information about that artist. Because of this, some of the famous artists like **Monet** were penalised. The results do not imply that **Monet** is less famous than the other artists, but they just indicate that our sources of information did not have much information about **Monet**. Unfortunately, we could not really do much about it since there is no site on the web which contains a comprehensive list of paintings.

This reminds us once again about the nature of the web, basically that the information is rarely stored in one place, but it is distributed. This is definitely not a big problem since in all the experiments presented in this document, our approach manages to overcome this problem. If we now take a look at the IE results, it still manages to find new paintings for every artist even though the paintings which are used as seed (i.e. those that were found by using II) are few in some cases. If we keep on looking at the results we got for **Monet**, the recall increases by 132% when the IE techniques are used.

On the whole the methodology starting only from the name of the artist manages to find quite a big list of paintings for each artist. In all cases II and IE combined together proved to be the way forward to gather information from the web effectively.

5.4.4 Analysis and Observations

In summary, when we analyse all the results, some interesting trends emerge. When the modules make use of II techniques, they all manage to get a high precision and a low recall. When IE techniques are used on top of the II techniques, the precision in all cases is stable (i.e. it either drops or rises by a slight amount or else it stays the

same) while the recall increases drastically. Therefore, whenever our methodology is fully applied, we manage to obtain both high precision and recall. Since this pattern was constant through out all the experiments, we conclude that it is the default modus operandi of how our methodology behaves. These results are important because they show that our system manages to strike a high balance between both precision and recall. The results obtained in most of the IE systems mentioned in Chapter 3 all manage to get high precision but low recall (or vice-versa). Our methodology on the other hand shows that if II and IE techniques are combined together, we can easily obtain high recall and precision.

The rationale behind our results is the following. With II techniques, we manage to obtain high precision but low recall. The results exhibit this trend because the several sources found on the web, no matter how updated they are cannot possibly manage to be as up to date as the place from where the data originates. Therefore with II we manage to get a high level of precision because:

- The sites contain next to perfect information and therefore we are confident that the data we gather from these sites is generally correct. We are not guaranteed it is the most up to date, but we are guaranteed its correctness since these sites are specialised in providing such data.

- These sites contain information presented in a structured way. Therefore, it makes the process of extracting information from these sites much easier and less error prone. Because of this, we eliminate lots of noise from our data immediately.

The only problem with II techniques is that the information obtained from the different sources is generally not the latest piece of data. In our methodology, we are

interested in harvesting the latest information from the web. When we test the results (which we obtain from the II techniques), most of the data is correct (therefore the precision is high) but since it is not the latest, the recall appears to be low in most cases since our data does not include the most recent information.

At this stage IE comes to the rescue, it allows us to bring up the recall of the system. First of all we make use of the precise examples (which we found using the II techniques) to find web pages where these examples appear. Our rationale here is that if these examples appear in a page, it is likely that the page contains other unseen examples. Once we have these pages, we mark the examples which we already know, get these examples together with some context and send these phrases to the IE algorithm for training. The trained IE system is then reapplied on the same page with the hope of discovering some new examples. Since the system is learning from good information, it is very unlikely that noise is introduced therefore precision is very stable. Recall shoots up because the page from which we are extracting information, is the page, which contains the latest information. In most cases, the pages returned by the search engines are the original pages written by the owner of the information therefore they are the most up to date. In synthesis, the IE succeeds because it learns the page structure and exploits that information to gather more information found in the page.

The conditions for our methodology to work are very simple, basically the system only needs an external partial source of information which helps the system spot precise information (but which must not necessarily be complete). Once this information is available it is used to seed the IE process, bootstrap learning and make the discovery of new instances possible.

5.5 Conclusion

In the Armadillo methodology, we have shown how to exploit two features of the web, the redundancy of information and the availability of information. Redundancy is an important feature of the data found on the web and refers to the property that the same information is not stored in just one place but is repeated in different repositories around the web. The availability of information discussed in (14) refers to the fact that online information can be used to bootstrap this process but in some cases, this information is not available online for a number of reasons. These include cases where the information is expected to be known by everyone and therefore no one bothers to publish it online. There are many other domains which suffer from the same problem of lack of published information and so far there are few plausible ways of tackling it apart from this methodology.

Armadillo is far from solving all the open issues related to the annotation task on the web, but the results clearly show that what we propose, is definitely as step forward in the right direction. To conclude, in the coming chapter we will summarise the research found in this thesis, share our vision of the future of the SW and we will propose new (sometimes daring) ideas. In so doing, we hope to stimulate more research in the SW field.

Chapter 6

Conclusion

6.1 Research Contributions

Throughout this thesis, we have seen how vital the whole annotation process is for the success of the Semantic Web (SW). All SW researchers understand the potential of this new web, they all expect great things to happen in the near future but most of them seem to put aside the annotation problem. As can be seen from the SW Wave diagram[1](See Figure 6.1), these tags are the building blocks upon which the SW idea is built. At the bottom, we find the markup languages (SGML, XML, etc.) and on top of them, all the other layers starting from resources and proceeding up to trust. Unfortunately the annotation task is far from being trivial since it is tedious, error prone and difficult when performed by humans. Apart from this, the number of web pages on the internet is increasing drastically every second, therefore making it unfeasible to have someone manually annotating all these pages. We could consider asking the authors of the web sites to include semantic annotations inside their own documents but would not this be similar to asking users to index their own pages? How many users would go about doing so?

---

[1]http://www.w3.org/2003/Talks/01-siia-tbl/slide19-0.html

Figure 6.1: The Semantic Web "Wave" as presented by Tim Berners-Lee.

Some years ago, pressure was made towards inserting meta tags inside HTML documents. These tags insert information into the header of web pages. The information is not visible by those viewing the pages in browsers. Instead, meta information in this area is used to communicate information that a human visitor may not be concerned with, for example, it can tell a browser what "character set" to use or whether a web page has self-rated itself in terms of adult content. These tags laid the initial steps towards inserting semantic information[2] into a web page. Since the users could not really understand the benefit they get from inserting these tags and considering search engines were not giving them particular weight, the meta tag suffered a quiet death.

The web reached this massive popularity and exponential growth mainly due to sociological factors. It was a structure originally created by people for people. Since people are social animals, they have an innate desire to socialise, take part in a community and make themselves known in that community. Also, in the fast moving world of today, we live in a society where people want results immediately and they expect immediate gratification for their efforts. The web offers just that!

The original web consisted of a network of academics. These people all came from the same community. They needed to get to know other people working in their area, share their knowledge and experiences, and also collaborate together. Snail Mail [3] was the only form of communication available between these academics but it was too slow. E-mail changed all this and made it possible for academics working far away

---

[2]The meta keywords tag allows the user to provide additional text for crawler-based search engines in order to index the keywords along with the body of the document. Unfortunately, for major search engines, it does not help at all since most crawlers now ignore the tag.

[3]A slang term, used to refer to traditional or surface mail sent through postal services. Nicknamed snail mail because the delivery time of a posted letter is slow when compared to the fast delivery of e-mail.

to exchange information in a few seconds. These academics saw a potential in this technology and therefore decided to grant access to this service even to their students. The latter saw even greater potential in this network and started experimenting with new ideas such as Bulletin Board Services (BBS) and online games (such as Multi User Dungeons (MUDs)). The original scope of the net was changed completely. It was not only limited to collaboration between individuals for work related purpose but became also a form of entertainment. This new and exciting technology could not be concealed for long within the university walls and quickly, people from outside these communities started using it too. It was a quick way of communicating with people, obtaining multimedia files, playing games etc. This web grew mainly because, it gained popularity in a closed community which was very influential and which gave it a very strong basis from where to expand. Subsequently, since this technology is easy to use by anyone and requires no special skills, its usage expanded at a quick rate.

Even though all this happened before the web as we know it today was even conceived, the current web grew exactly in the same way. With the appearance of web browsers, people who have been using the internet realised that they could move away from the dull world of textual interfaces and express themselves using pages containing rich multimedia content. People soon realised that it was not just a way of presenting their work, this new media gave them an opportunity even to present themselves to others. It allowed anyone to have his/her own corner on the internet accessible to everyone else. A sort of online showcase which was always there representing the personal views, ideas, dreams etc twenty four hours a day, seven days a week! The technology to create these pages was a little bit difficult to grasp

initially but soon became extremely user friendly having editors very similar to text processors (a technology which has been around for decades and which was known by everyone who was computer literate). To cut a long story short, everybody wanted to be present in this online world. Most people did not even know why they wanted to be there!

The initial popularity grew mainly due to word of mouth, but that growth is insignificant when compared with the current expansion which the web is experiencing. Search engines were quite vital for this success. Before search engines were conceived, one knew where to find information only by asking other people or communities (which were specialised in the area) regarding sites where information could be found. After, the process of looking for information was just a matter of using a search engine. Therefore, information did not have to be advertised anywhere since automatic programs called crawlers were capable of searching the web for information and keeping large indexes with details of where that information is found.

The creator of the SW (Tim Berners-Lee who happens to be also the creator of the web) is trying to make the SW a reality using the same approach. I.e. the SW should not be something created by a group of academics for people, but it should emerge as being something made by people for people. For this to happen we should base our work upon previous experiences learnt when creating the current web. In the same way as search engines are vital for the web of today, semantic search engines will be vital for the web of tomorrow. These search engines will allow searches to be conducted using semantics rather than using the bag of words approach (popular in today's search engines). For this to happen, we must have programs similar to crawlers that create semantic indexes containing references to

documents using semantics, rather than keywords. To discover these semantics, we further require some automatic semantic annotators which semantically annotate the documents. In this thesis, we tackled exactly this task by proposing two annotating methodologies.

The first one is semiautomatic and makes use of the interaction between user, interface and Information Extraction System (IES) for human-centred document annotation that both maximises timeliness and minimises intrusiveness of the IES suggestions. An important feature of this methodology is that it manages to empower users to maximally customise the degree of proactivity so as to obtain the desired behaviour from the IES. Originally, it was intended as a prototype system designed to show case how Information Extraction (IE) can be utilised efficiently and effectively especially during the training phase. Melita is an annotation interface that actively supports interaction between users and an adaptive IE system. The idea is that while users tag documents for training the IE system, Melita

1. avoids the user re-annotating examples for which the IE system is already able to annotate.

2. helps selecting the more informative documents to be annotated (to maximise effectiveness of the training process)

To do that, Melita runs in the background an adaptive IE system and provides timely annotation of already covered cases. The selection of documents for annotation is also based on an analysis of the IE system's performance on an un-tagged corpus. This methodology has proved to be suitable and effective whenever a closed set of documents need to be annotated. In the experiments conducted, we have shown that the IES provides effective high quality annotation with little effort.

The second methodology is similar to the first one but the human factor is virtually eliminated by making use of Information Extraction (IE) and Information Integration (II) technologies. Information is initially extracted by starting from highly reliable/easy-to-mine sources such as databases and digital libraries and is then used to bootstrap more complex modules such as wrappers for extracting information from highly regular Web pages. Information extracted by the wrappers is then used to train more sophisticated IE engines. All the annotated training corpora for the IE engines is produced automatically. Experimental results show that the methodology can produce high quality results.

The user intervention is limited to provide an initial element of information in order to instruct the engine where to start searching for the desired data and to act as an Oracle by adding information, missed by the different modules, when the computation is finished. No preliminary manual annotation is required. The information added or deleted by the user can then be reused for restarting learning and therefore getting more information (recall) and/or more precision. The type of user needed is a person able to understand the annotation task. No skills in IE are needed.

The natural application of such a methodology is the Web, but large companies' repositories are also an option. In this document we have focused on the use of the technology for mining web sites, an issue that can become very relevant for the Semantic Web, especially because annotation is provided largely without user intervention. It could potentially provide a partial solution to the outstanding problem of who is providing semantic annotation for the SW. It can potentially be used either by search engines associated to services/ontologies to automatically annotate/index/retrieve relevant documents or by specific users to retrieve the required information on the

fly.

The idea of using the redundancy of information to bootstrap IE learning is not new, having been already proposed by Brin (15) and Mitchell (87). The difference with our approach is the way in which learning is bootstrapped. Brin uses user-defined examples, while Mitchell uses generic patterns that work independently from the place at hand (e.g. the site or the page). We use both the above[4], but in addition, we exploit the redundancy of information and integrate information extracted from different sources with different levels of complexity. In this respect our approach is to our knowledge unique.

As noted by Brin, great care is needed in order to select only reliable information for annotation for learning. The integration of different knowledge sources multiplies the available information, and therefore allows learning to be seeded only when multiple evidence is found. The experimental results described in this document were obtained by fixing high precision ($>90\%$) and testing the obtained recall. One relevant question for the effective usability of the methodology in real applications concerns the required level of accuracy (as a balance of precision and recall) the system has to provide. As Web applications are concerned, it is well known that high accuracy is not always required. Search engines are used every day by millions of people, even if their accuracy is far from ideal: further navigation is often required to find satisfying results, large portions of the Web are not indexed (the so called dark and invisible Webs), etc. Services like Citeseer, although incomplete, are very successful. What really seems to matter is size: the ability to both retrieve information dispersed on the Web and create a critical mass or relatively reliable information. In this respect

---

[4]Generic patterns are used in the named entity recogniser.

the proposed methodology is satisfying. Experiments show that in the case of paper discovery, it is able to discover a large part of the information that the digital libraries (which were used to seed) did not have (+50%). Precision is topping 90%, so the information provided is very reliable. This methodology was implemented in the Armadillo system.

Both systems proved to be so useful that they opened various doors into several other fields of research such a Human Computer Interaction, Visualisation and Adaptive Information Extraction. The systems were also distributed to a number of Universities and private companies for evaluation purposes. They have both gained national and international recognition. In fact, the Melita system has been evaluated by a British Company with the intent of commercialising it and Armadillo was a core component of the CS AKTive Space (99) (100) (59) application that won first prize in the 2003 Semantic Web challenge[5]. Both applications were also used as demonstration tools in various international Summer Schools. They were also rated world class in a mid-term review report created for the AKT project[6] by a panel of international experts whose chair was Professor James Hendler (one of the creators of the Semantic Web). Although, these methodologies do not manage to solve all the problems associated with the SW, they definitely contribute a step forward in the right direction towards making the SW a reality!

6.2 Future Directions

This section takes a look at what we believe to be the future directions where SW related research should go. The first sections highlight what problems IE and II face

---

[5]http://www-agki.tzi.de/swc/

[6]http://www.aktors.org/

when confronted with the web. The final sections will present several potential future enhancements and usages of the methodologies developed in this document. They will propose different ways in which annotations will change from how we know them today and what their new role will be in the big SW picture.

6.2.1 IE and the web

From a semantic point of view, the web of today is not very different from the SW of the future. In fact it already contains a lot of information, but the only problem is that it is targeted for human consumption. Machines are not capable of interpreting this data in order to process it further. Thus, the need arises to annotate all the information found on the internet using tags which are understandable both by humans and machines.

As we have already seen, manual annotation is difficult and time consuming to perform. The resources which such an exercise would require are unimaginable. If we make use of a semi or fully automatic methodology (like the ones presented in this thesis), the task would be simplified by many orders of magnitude, but can we do better than that?

An important component of our approach is IE. IE algorithms are first trained to identify the information which is important for us. Then they automatically extract similar information from other unseen documents. Unfortunately, the results we obtain are not always very good and sometimes, they are very much dependent upon the complexity of the page being processed. So far, most of the successful algorithms focused on linguistic approaches and very few of them took into consideration properties associated with the document structure, layout, etc. Basically, most of them were ignoring the visual properties of a document, that is why we are proposing a

new breed of IE tools called Visual IE (VIE).

VIE is not something new, the most advanced information processing system (the human being) has been using it for thousands of years, so why not replicate it? The brain is a fantastic organ which makes use of associative memory whereby different elements are associated together. These elements can be of various types, they can be linguistic, visual or an infinite number of other types. Like a big jigsaw puzzle, all these associations in the brain make it possible for a person to relate several pieces together and in so doing, groupings start to emerge until finally, the big picture becomes clear. The huge number of associations also makes it possible for the whole process to cater for noisy environments. If we take a look at the text "50ME1 GR8", a machine would not be capable of interpreting it whilst a human can easily understand that it means "Someone great". This trick is currently being used by spammers all over the internet to fool automated email filters and so far, we must admit that they are succeeding in doing so!

Our VIE system does not necessarily has to be as sophisticated as the human brain, but at least we should start taking visual cues into consideration. The VIE must be taught the effect of specific formatting, not just visually, but also semantically. For example, a text in **bold** or *italics* might signify a certain degree of importance with respect to the other text in the document. Another important visual cue is proximity, it refers to the fact that elements close to each other might be related together in some way. A piece of text which is closer to an image than to other elements in the document might be the caption of the image. Some text which is formatted as being important at the top of the document might be the title of the document. The list of examples can keep on going on forever. If we abstract further, a group of words in

the proximity of each other form a paragraph. And what is a paragraph if not a small set of ideas which when combined with other related paragraphs form a document? What about other complex structures such as tables and lists, aren't they a grouping of related data as well?

There is an infinite number of visual cues one can learn to extract from a document. To cover them all, one would require another thesis solely dedicated to this subject. The message we would like to convey is that the information already exists on the web, there's no need of annotating everything. The information is there waiting for us to harvest it, and only with tools like VIE we can manage to do so!

6.2.2   IE and the SW challenges

From the IE point of view with regards to the SW, there are a number of challenges in learning from automatic annotation, instead of using human annotation. On the one hand not all human annotation is reliable: the use of multiple strategies and combined evidence reduces the problem, but still there is a strong need for methodologies robust with respect to noise. On the other hand, many IE systems are able to learn from completely annotated documents only, so that all the annotated strings are considered positive examples and the rest of the text is used as a set of counterexamples. In our cycle of seed and learn, we generally produce partially annotated documents. This means that the system is presented with positive examples, but the rest of the texts can never be considered as a set of negative examples, because un-annotated portions of text can contain instances that the system has to discover, not counterexamples.

This is a challenge for the learner. At the moment we present the learner with just the annotated portion of the text plus a windows of words of context, not with

the whole document. This is enough to have the system learning correctly: the un-annotated examples that become negative examples entering the training corpus is generally low enough to avoid problems. In the future we will have to focus on using machine learning methodologies that are able to learn from scattered annotation.

6.2.3 II and the SW challenges

The methodology proposed in this document is based on using the redundancy of information. Information is extracted from different sources (databases, digital libraries, documents, etc.), therefore the classic problems of integrating information arise. Information can be represented in different ways in different sources from both a syntactic and a semantic point of view.

The syntactic variation is coped with when we define the architecture because when two modules are connected, a canonical form of the information is defined, e.g. the classic problem of recognising film titles as "The big chill" and "Big chill, the" can be addressed. More complex tasks need to be addressed, though. For example, a person name can be cited in different ways: J. Smith, John Smith and John William Smith are potential variation of the same name. But do they identify the same person as well? When a large quantity of information is available (e.g. authors names in Citeseer) this becomes an important issue (4). This problem intersects with that of intra- and inter-document co-reference resolution well known in Natural Language Processing.

We are currently focusing on mining websites, because this allows to apply some heuristics that very often solve these problems in a satisfying way. For example the probability that J. Smith, John Smith and John William Smith are not the same person in a specific CS website is very low and therefore it is possible to hypothesise

co-reference. Different is the case of ambiguity in the external resources (e.g. in the digital libraries). Here the problem is more pervasive. When querying with very common names like the example above, the results is quite disappointing, as papers by different people are mixed. This is not a problem in our approach because the information returned is used to annotate the site. Papers from people from other departments or universities will not introduce any annotations and therefore will not cause any problems. The same applies in case multiple home pages are returned: if they do not have an address local to the current site, the page is not used. In the generic case, though, this is a problem.

We live in a fast changing world full of discrepancies which are accepted by everyone. If we image a task where we need to integrate information concerning number of people living in a country, it is very likely that different reliable sources will have different values, therefore creating discrepancies. These values might be correct if they present a number which is approximately equal to a common average, accepted by everyone (but which is not necessarily the exact number!).

6.2.4 Armadillo: Agents over the Grid

Armadillo is primarily a web based system aimed at automatically harvesting information from the web. Although it serves its intended purpose quite well, it suffers from a number of limitations.

Exploitation of Resources Resources are limited and if they are not managed well, we risk wasting some of them. The internal modules of the current system work on a single machine. They do not exploit other machines available on a network. If these computers are available, it would be useful to distribute the internal

modules over several machines and execute them in parallel.

Coordination The distributed nature of the system is still coordinated by a centralised module. Although there is nothing wrong with this approach since most systems (even in the physical world) have a centralised control, in itself, control imposes additional overheads over the whole system. This module wastes a lot of precious processing power which can be used to perform further processing.

Robustness Since the system connects directly to external resources (such as web services), if these resources are not available, the whole process reaches a stalling point especially if the resource provides a critical function. These resources are not intelligent enough to cater for failures.

Accountability A module is not accountable for the data it produces. Although, every item of data is signed by the module which produces the data, if a module produces spurious data, there is nothing which stops the module or which calibrates it in order to produce correct data. All the data verification must be performed manually by the user.

To overcome these limitations, the system can easily be ported towards an agent[7] based architecture which exploits new technologies in distributed computing such as grid[8] computing. Let us take a look at how Agents and Grid Technology can overcome the different limitations of Armadillo.

---

[7]A computer program that carries out tasks on behalf of another entity. Frequently used to reference a program that searches the Internet for information meeting the specified requirements of an individual user.

[8]A grid is a software framework providing layers of services to access and manage distributed hardware and software resources.

In any computerised system, resources (such as processing power, secondary storage, main memory, etc.) are always scarce. No matter how much we have available, a system could always make use of more. Systems like Armadillo are very demanding, much more than normal ones. The system needs to process huge amounts of texts using complex linguistic tools in the least possible time. These demands have taken Armadillo away from the normal desktop platform found in every home, towards top of the range systems (and even servers). Physical resources are not the only bottleneck in the whole system. The raw material which the system utilises are web pages, downloaded from the internet. Unfortunately, this process is still extremely slow, especially when several pages are requested simultaneously from the same server. In synthesis, we can conclude that such systems are only usable by top of the range computers having a high bandwidth connection to the internet ... or is it?

Agents and Grid technology can make all the difference by efficiently **exploiting the resources we have to the maximum**. To achieve this, first of all we need to create an environment where agents can live accessible from anywhere over the internet. This would be like a virtual city, accessible over HTTP where agents can decide to go and live or work for some time. This city is basically just a server in which mobile agents can upload themselves freely. This community would have some Immigration Control Agents which monitor upload requests and they can:

Refuse entry to an agent based on some predefined heuristics (Eg Agent is black listed).

Grant an entry visa which allows an agent to exist in the city but with very little resources allotted to it. In this way, the agent can do simple processes like looking for another community, contact its creator for further instructions, etc.

Under this visa, the agent is expelled from the system after a certain period by either sending it to another community which is accepting agents or just deleting it.

Grant a work permit which allows an agent to exist in the city using as much resources as he can get from the city. This kind of visa is not bound by a time limit.

Once an agent is inside the city, it can utilise the resources of the city such as processing power, memory, etc. A city could also act as a portal to a grid network. This would allow agents to execute functions over the grid transparently without altering anything in the agents. The system would take care of distributing work over the grid. Other services must also be created for this community to work such as search engines which return links to cities together with related statistics (regarding as agent population, resource utilisation, etc). A city can be installed on any machine, it can be configured to use as little or as much resources are available. As for Armadillo, it is already divided into several modules, some internal and others external. All of these modules should be packaged inside an intelligent agent for the architecture to work.

The benefits of this system are various. Since the system is made up of several agents, on the client side, there is no more the need of a powerful machine with huge network bandwidth. The client is just a thin system whereby the user defines the strategies (As is the case in the current version of Armadillo). When the user is happy with the strategy, the system is initialised. Agents are created for every module used in the strategy and sent to a city somewhere over the internet for execution. Over here we are exploiting the property that an agent is able to move from one system to

another and through various architectures and platforms, also referred to as Agents Mobility. If no cities are found then they are executed locally.

A legitimate question to ask is, how would agents go about **coordinating their activities**? From our previous experiences with the Armadillo system, a data driven approach was enough to coordinate everything. This means that the data of the system is stored somewhere centrally, within reach of every module and each new data item inserted in the database acts as a trigger for a module to start execution. Every module had at least an input and an output. The data passed between modules was all semantically typed, therefore it was possible to have hooks connected to specific fields in the database which executed functions in a module whenever new data was available. The same can be applied to distributed agents. All we need is an agent that acts as a gatekeeper for the database and another agent used to wrap every module in the system. Agents can do this easily since they are:

- capable of interacting and cooperating with other agents or human beings in order to complete a task. Therefore the gatekeeper agent could easily ask another agent to execute a particular function whenever new data is available.

- capable of choosing what it will do, and in which order, according to the external environment. Since an agent can easily be overloaded with work, it can clone itself, send the other copies to other server and instruct them to perform a subtask of his work. A typical example would be an agent used to download a list of web pages. The agent could easily, clone itself and send the cloned agents to download a sublist of the whole list.

Another issue to consider is **robustness**. In a system which makes heavy use of the web, it is very common to try to access links or resources that are no longer

available. There can be various reasons for this, a server could be down, a page does not exist any longer, etc. Agents come to the rescue once again. Since agents are goal-oriented, they accept precise "human requests" and decide how to satisfy them. If an external resource which they require is not available they are able to take initiatives in an autonomous way and they can exert control over their own actions. In other words, they are capable of defining their own objectives and achieving them without human supervision. Lets imagine an agent requires an online resource which is not available, there are several strategies it could take. The most intelligent agent would look for another online resource as a substitute (But probably we are expecting too much from them!). A more realistic view is to contact the client and ask for the link to another resource. Another approach which an agent might take is to go to sleep and try to access the service later. It is quite common on the internet to have a server down for some time. Agents can easily do this because contrary to traditional software, an agent can decide to start an action at a precise moment, based on the state of the external environment.

The last weakness of the system is **accountability**. Whenever new data is produced agents should add to this data some meta information such as which agent created this data, at what time and from which source it obtained the data. Agents should be able to adapt to the users needs by analysing its past actions. Therefore, if an item of data was reviewed by a user and rejected, the agent should reevaluate the reliability rating of the source from where the data was obtained. The global reliability rating of a site must be a reflection of the data it produces.

The system should also be capable of performing automatic checks without the user's intervention and deciding what appropriate action to take when necessary.

Imagine some seed data is used to train a learning algorithm. If the precision of the algorithm is low when tested on the training data itself, then the agent should realise that the learning algorithm is not good enough and exclude it from the whole process automatically. This is just one of the many automatic tests which can be performed by agents. The degree of automation of these tests depends entirely upon the application and its complexity.

6.2.5 The Semantic Annotation Engine

A Semantic Annotation Engine is a engine that given a document annotates it with semantic tags. The need for such an engine arises from the fact that the web currently contains millions of documents and its size is constantly increasing(30). When we are faced with such huge tasks, it is inconceivable that humans will manage to annotate such a huge amount of documents. An alternative would be to use automated tools such as Armadillo (Refer to Chapter 5). Armadillo makes use of the information present on the web combined with IE and II techniques in order to discover new facts in the document. But this is a partial solution which also is not very feasible even though it would be more successful than manual annotation for sure since the system would not get bored of annotating huge amounts of documents. There are a number of problems to face with this approach. First of all the system is specific to a particular domain and cannot just semantically annotate any document (especially if that document is not related to the current domain). If we imagine that the system could annotate any document, a problem arises regarding which annotation is required for which document. In theory a document could have an infinite number of annotations because different users would need different views of a particular document. Therefore, deciding which document contains which annotations should

not be a task assigned to the annotation system but rather to the user requesting the annotations. Apart from this, there are still some open issues with regards to annotations per se, like should annotations be inserted as soon as a document is created or not? Some annotations can become out of date very quickly like weather reports, stock exchange rates, positions inside a company, etc. What would happen when this volatile information changes? Would we have to re-annotate the documents again? But that would mean that we would not need annotation engines but re-annotating engines constantly maintaining annotations. The order of magnitude of the problem would therefore increase exponentially. The second problem is that probably, the rate at which the web is growing is much faster than such a system could annotate. We would therefore require a small army of Armadillo's to try to catch up with such a growth.

Another more realistic and generic solution than Armadillo would be to provide annotations on demand instead of pre-annotating all the documents i.e. the Semantic Annotation Engine (SAE). This means that annotations will always be the most recent and there would not be any legacy with the past. If a document is out of date or updated with more recent information, the annotations would still be the most recent. The system would make use of currently available technologies but in a smarter way as can be seen in Figure 6.2.

In traditional Information Retrieval (IR) engines (like Google[9], Yahoo[10], etc) a collection of documents (such as the web) is crawled and indexed. Queries using a bag of words are used to retrieve the page within the collection that contains most (if not all) of the words in the query. These engines do a pretty good job at indexing

---

[9]http://www.google.com
[10]http://www.yahoo.com

a substantial part of the web and retrieving relevant documents, but they are still far from providing the user exactly what he needs. Most of the time, the users must filter the results returned by the search engine. The SAE would not affect in any way the current IR setup. The search through all the documents would still be performed using the traditional IR methods, the only difference would be when the engine returns back to the user the results. Instead of passing the results back to the user, they are passed to a SAE. As can be seen from Figure 6.2, the system contains an index of the different SAEs which are available online. The SAEs are indexed using keywords similarly to how normal indexing of web documents works. Whenever an IR engine receives a query, it not only retrieves the best document with the highest relevance but also the SAEs with the highest relevance. These are then used to index the documents retrieved by the search engine before passing them back (annotated) either to the user or to an intermediate system that performs further processing. SAE will provide on the fly annotations for web documents therefore avoiding the need of annotating the millions of documents on the web beforehand.

One relevant question for the effective usability of this methodology in real applications concerns the required level of accuracy (as a balance of precision and recall) the system has to provide. As Web applications are concerned, it is well known that high accuracy is not always required. Search engines are used every day by millions of people, even if their accuracy is far from ideal: further navigation is often required to find satisfying results, large portions of the Web are not indexed (the so called dark and invisible Webs), etc. Services like Citeseer, although incomplete, are very successful. What really seems to matter is size: the ability to both retrieve information dispersed on the Web and create a critical mass or relatively reliable information.
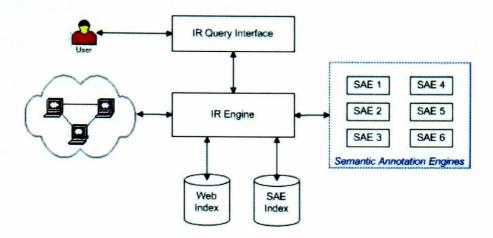
Figure 6.2: A potential architecture for a Semantic Annotation Engine.

6.2.6 The Semantic Web Proxy

A Semantic Web Proxy (SWP) is similar to a normal web proxy[11], it provides all the functionality associated with such a program. The main difference is that it also provides some semantic functions hidden from the user.

To understand what these semantic functions are, lets take a look at a typical example. Imagine a user would like to purchase a computer having a 17 inch flat screen monitor, 1 GBytes of RAM, a 3 GHz processor, etc. The user would go to an online store, look for a search form where he can write the search criteria and perform the search for the desired product. This operation must be repeated for each and every online store he would like to query.

A SWP would simplify this process in several ways. First of all, it would keep a record of all the forms being filled in a single session [12] and the content of these

---

[11]A proxy is an intermediate server that sits between the client and the origin server. It accepts requests from clients, transmits those requests on to the origin server, and then returns the response from the origin server to the client. If several clients request the same content, the proxy can deliver that content from its cache, rather than requesting it from the origin server each time, thereby reducing response time.

[12]A session is a series of transactions or hits made by a single user. If there has been no activity for a period of time, followed by the resumption of activity by the same user, a new session is considered

forms. If the forms are semantically tagged or associated with an ontology, then the details inserted by the user are automatically tagged as well. Once ready, the user would then go to a different site and perform a similar search. The system would notice that the form in the new site is filled with the same details as those in the other site and it would take the following actions:

1. tag the details inserted by the user with the semantic tags found in the new site.

2. create equivalence relationships between the semantic tags in the first site and the tags in the second site.

3. make these relationships available to everyone.

The effect of this process is the following. If another user performs a search for a similar item in one of the sites already in the SWP, the system uses the shared relationships (obtained before through the interaction with the other users) and automatically searches the other online stores. All the results are returned to the user for evaluation.

The system basically creates links between ontologies in a shared way collaboratively without the user realising it. It does so by examining the browsing habits of the user and deducing implicit relations when possible. Once again, the system adopts the same ideas used to create the web. Basically, it exploits the little work of many users (without adding any extra effort) to automatically create relationships between concepts over the web.

---

started.

6.2.7   Semantic Plugins

Semantic Plugins (SP) are application addons which can be inserted to any application having two major tasks. First of all , they must harvest semantic information from the user without the user realising it. This information must be stored in ways which can be reused by other applications. Secondly, they must provide a value added service to the user without increasing the effort of using the application. If they do increase a little the effort required, they must make sure that the return is worth the extra effort. This second requirement is important because as we have already seen in Section 6.1 (with the example of the Meta Tag), if a user does not get an immediate return from using a new technology, then no one will go through the learning curve involved to use that technology. As an example of a SP, imagine a normal word processor enhanced with a SP aimed specifically for paper writing.

If we look at the process involved in writing a paper (or even any other document) we immediately realise that it is quite an unnatural process. People just dump down their thoughts in an ordered way and then they must further structure the text making sure their ideas are coherent and finally, gather external evidence and supporting statements from other documents. All this makes the whole process quite difficult to perform unless the author is a skilled writer well trained in the art of writing. Unfortunately, this writing methodology is most probably far from the way human thoughts are structured. Information is not stored in the brain as a big structured document but rather as small atomic and composite concepts attached together by several relationships. The idea we propose here is not to drastically change the way humans have been writing for thousands of years, but rather to start moving slowly towards this new model of writing which is slightly more natural than traditional

methods. Whilst doing so, we must not forget our main aim, i.e. provide tools in normal text editors which allow people to transparently insert semantics in document without themselves knowing it.

Let us look at a scenario where a user is writing a paper. The process involves writing the paper and adding references from papers (which the author previously read). So basically, the paper is a self contained piece of knowledge which has no physical links to anything else. There are only implicit semantic links expressed in the form of references and one need a powerful processor like a human or some clever code in order to try to find which link refers to what. The SP would be a program running in the background which analyses what its being written. It grasps a sentence (could be a paragraph or phrases too) at a time and compares them with papers already published. When it finds similar sentences, it shows the extracts to the user in a continuously updated list. The user sees this list and he can read the extract and insert a reference where he is writing by just clicking on the link. Adding references to a paper is a must do for every user, therefore this program will make the user's life easier. There is no need for the user to dig in his memories trying to remember in which paper he read about a particular concept many months ago. This program will also add value to his paper. If there is a relevant paper which the user did not read, the program might be able to find it for him.

Contrary to what is happening today, a paper produced with the help of the SP is not a self contained piece of knowledge but it should contain links to semantic information about the references. These links could point to a location inside the same document or to a URI somewhere over the Internet. In synthesis, it should be semantically annotated with tags stating that a particular concept in the document is

the same as the concept defined in the paper being referenced. The relationships can be created automatically by the system since each time a user inserts a reference, he is implicitly creating a semantic link between that portion of the text and the contents of the reference. What we are doing differently here is that the SP is automatically explicitly defining that relationship in the document.

This allows the system to associate pieces of text inside one document with other pieces of text inside another document. All the information provided by the user will be kept. So another system could further infer with a very high degree of certainty that line X in document A is associated with line Y in document B. The parts not associated with any other documents are probably original parts referring to the new work in the document per-se. The same should be done for picture references, table references etc. They are not objects inserted in the document at random. They normally would have a caption associated with them and apart from that, they are mentioned somewhere in the document as well.

The document could be further analysed with NLP techniques such as co-reference resolution, relational IE etc in order to make it easier to find new facts. Also, since the SP in this domain would know things like authors etc, it could see previous publications of the same author(s) and find similarities between them. In this way it could find inter-document connections of documents written by the same authors. These connections might help establish the section of text which describes the innovative part from one document to another. I.e. the intersection between a new document and an old document is the innovative part of the new document.

Documents could also be indexed semantically. The system would go about achieving this by guiding the user into writing a document using a new perspective. Instead

of writing a whole document, the user could be asked to first write a storyline, made up of just points and then gradually expanding and restructuring the order of those points. How the document evolves from a set of points to sentences and then paragraphs is all stored since every step is a refinement of the previous one. Therefore these refinements could be used to semantically index the document. This kind of indexing would offer lots of benefits. Lets imagine that some months later, the user is compiling a report using these documents. The typical process involves going through the written documents looking for the paragraphs required for the new document. If instead of this, we have a semantic index which stores different granularities of the same idea (using the refinement technique mentioned before), the user could just look for specific paragraphs using their semantic content as search terms. The system would then insert the pieces of text inside the new document and the job of the user is just to glue the different pieces together. Some smart grammatical analysis can also be done at this stage to propose a possible glue. Basically, the job of the user would just be to check the document and correct it. The portion of text extracted from the other documents could either be a copy of the text or just a link to the other document. In this way, if the text in the original document is updated, the text in the new document is updated automatically.

SPs propose a new way of interacting with the user which does not add any extra burden on the user's behalf, but which adds value to

the user since the discovery of references is facilitated and the writing of a document is easier.

the document since the document would contain semantic information stored inside it. This information can be further used for a variety of applications like

automatically compiling documents from other documents using just a guideline by the user, making semantic search engines a reality since the engines will index sub parts of a document semantically, etc.

6.3 Summary

In summary, we have presented two methodologies for annotating documents, one semi-automatic and the other fully automatic. They make use of several Artificial Intelligence techniques such as Natural Language Processing (NLP) and Machine Learning (ML). The combination of these techniques contributes towards creating systems capable of advanced text analysis that manage to discover which parts of the text to annotate and which can be easily adapted to new domains. We believe that these technologies are effective enough to support the annotation process which is a requirement of the SW. This smart combination of techniques made it possible to achieve SW objectives which were unreachable before!

Bibliography

[1] *Badger information extraction (IE) software*, http://www-nlp.cs.umass.edu/software/badger.html.

[2] *Special issue on information integration*, vol. 41, IBM Systems Journal, no. 4, 2002.

[3] K Aditya, B Parsia, J Hendler, and J Golbeck, *SMORE - semantic markup, ontology, and RDF editor*.

[4] H Alani, S Dasmahapatra, N Gibbins, H Glaser, S Harris, Y Kalfoglou, K O'Hara, and N Shadbolt, *Managing reference: Ensuring referential integrity of ontologies for the semantic web*, Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02, Springer Verlag, 2002.

[5] H Alshawi, *The core language engine*, Cambridge, MA: MIT Press, 1992.

[6] J L Ambite, N Ashish, G Barish, C A Knoblock, S Minton, P J Modi, I Muslea, A Philpot, and S Tejada, *Ariadne: a system for constructing mediators for Internet sources*, 1998, pp. 561–563.

[7] R Basili, F Ciravegna, and R Gaizauskas (eds.), *Workshop on machine learning for ie*, Berlin, ECAI2000, 2000.

[8] E Bauer and R Kohavi, *An empirical comparison of voting classification algorithms: Bagging, boosting, and variants*, Machine Learning **36** (1999).

247

[9] A Berger, *A brief maximum entropy tutorial*, http://www-2.cs.cmu.edu/afs/cs/user/aberger/www/html/tutorial/tutorial.html, 1996.

[10] T Berners-Lee, J Hendler, and O Lassila, *The semantic web*, Scientific American (2001), 30–37.

[11] K Bharat, B W Chang, M R Henzinger, and M Ruhl, *Who links to whom: Mining linkage between web sites*, ICDM, 2001, pp. 51–58.

[12] P Borst, H Akkermans, and J Top, *Engineering ontologies*, Proceedings of the 10th Knowledge Acquisition for Knowledge Based Systems Workshop, 1996.

[13] L Breiman, *Bagging predicates*, 421, University of California, September 1994.

[14] C Brewster, F Ciravegna, and Y Wilks, *Background and foreground knowledge in dynamic ontology construction*, Proceedings of the Semantic Web Workshop, Toronto, August 2003, SIGIR, 2003.

[15] S Brin, *Extracting patterns and relations from the world wide web*, WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98, 1998.

[16] J Brown and E Gray, *The people are the company*, Fast Company 1:78-82. http://www.fastcompany.com/online/01/people.html, 1995.

[17] M E Califf, *Relational learning techniques for natural language extraction*, Tech. Report AI98-276, 1, 1998.

[18] M E Califf, D Freitag, N Kushmerick, and I Muslea (eds.), *Workshop on machine learning for information extraction*, Orlando, Florida, AAAI-99, July 1999.

[19] M E Califf and R J Mooney, *Relational learning of pattern-match rules for information extraction*, ACL Workshop on Natural Language Information Extraction, Association of Computer Linguists, July 1997.

[20] N Català, *Essence: a portable methodology for building information extraction systems*, 1998.

[21] N Català, N Castell, and M Martín, *Essence: a portable methodology for acquiring information extraction patterns*, 2000.

[22] H L Chieu and H T Ng, *A maximum entropy approach to information extraction from semi-structured and free text*, Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)., 2002, pp. 786–791.

[23] F Ciravegna, *Adaptive information extraction from text by rule induction and generalisation*, Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI), 2001, Seattle.

[24] ———, *(LP)², an adaptive algorithm for information extraction from web-related texts*, Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th International Joint Conference on Artificial Intelligence, 2001, Seattle, http://www.smi.ucd.ie/ATEM2001/.

[25] ———, *Designing adaptive information extraction for the semantic web in amilcare*, Annotation for the Semantic Web (S Handschuh and S Staab, eds.), Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdam, 2003.

[26] F Ciravegna, A Dingli, D Petrelli, and Y Wilks, *User-system cooperation in document annotation based on information extraction*, Proceedings of the 13th

International Conference on Knowledge Engineering and Knowledge Management, EKAW02, Springer Verlag, 2002.

[27] F Ciravegna, N Kushmenrick, R Mooney, and I Muslea (eds.), *Workshop on adaptive text extraction and mining held in conjunction with the 17th international conference on artificial intelligence*, Seattle, IJCAI-2001, August 2001.

[28] F Ciravegna and D Petrelli, *User involvement in adaptive information extraction: Position paper*, Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th International Joint Conference on Artificial Intelligence, 2001, Seattle, http://www.smi.ucd.ie/ATEM2001/.

[29] P Clark, J Thompson, H Holmback, and L Duncan, *Exploiting a thesaurus-based semantic net for knowledge-based search*, AAAI/IAAI, 2000, pp. 988–995.

[30] K Coffman and A Odlyzko, *The size and growth rate of the internet.*

[31] W Cohen, P Ravikumar, and S E Fienberg, *A comparison of string distance metrics for name-matching tasks*, 2003.

[32] W W Cohen, *Fast effective rule induction*, Proc. of the 12th International Conference on Machine Learning (Tahoe City, CA) (Armand Prieditis and Stuart Russell, eds.), Morgan Kaufmann, July 9–12, 1995, pp. 115–123.

[33] C Cumby and D Roth, *Relational representations that facilitate learning*, KR2000: Principles of Knowledge Representation and Reasoning (San Francisco) (Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, eds.), Morgan Kaufmann, 2000, pp. 425–434.

[34] D David, J Aberdeen, L Hirschman, R Kozierok, P Robinson, and M Vilain, *Mixed-initiative development of language processing systems*, Fifth Conference on Applied Natural Language Processing, April 1997, pp. 348–355.

[35] L Dehaspe, H Blockeel, and L De Raedt, *Induction, logic and natural language processing*, The joint ELSNET/COMPULOG-NET/EAGLES Workshop on Computational Logic for Natural Language Processing, 1995.

[36] L Denoue and L Vignollet, *An annotation tool for web browsers and its applications to information retrieval*, 2000.

[37] A Doan, Y Lu, Y Lee, and J Han, *Object matching for information integration: A profiler-based approach*, 2003.

[38] J B Domingue, M Lanzoni, E Motta, M Vargas-Vera, and F Ciravegna, *Mnm: Ontology driven semi-automatic or automatic support for semantic markup*, 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), October 2002.

[39] S Dumais, M Banko, E Brill, J Lin, and A Ng, *Web question answering: Is more always better?*, Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002) (Tampere, Finland), 2002.

[40] S Dzeroski, *Handling imperfect data in inductive logic programming*, Fourth Scandinavian Conference on Artificial Intelligence, IOS Press, 1993, pp. 111–125.

[41] S Engelson and I Dagan, *Minimizing manual annotation cost in supervised training from corpora*, 34th Annual Meeting of the ACL (ACL'96), 1996, pp. 319–326.

[42] Y Wilks F Ciravegna, A Dingli and D Petrelli, *Using adaptive information extraction for effective human-centred document annotation*, Text Mining, Theoretical Aspects and Applications (I.Renz J.Franke, G.Nakhaeizadeh, ed.), Lecture Notes in Computer Science, Springer Verlag, 2003.

[43] D Freitag, *Information extraction from html: Application of a general learning approach*, Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98 (1998), 517–523.

[44] _____, *Machine learning for information extraction in informal domains*, Ph.D. thesis, CMU, November 1998.

[45] _____, *Multistrategy learning for information extraction*, Proceedings of ICML-98 (1998).

[46] D Freitag and N Kushmerick, *Boosted wrapper induction*, American Association for Artificial Intelligence (AAAI00), 2000.

[47] _____, *Boosted wrapper induction*, ECAI2000 Workshop on Machine Learning for Information Extraction (R Basili, F Ciravegna, and R Gaizauskas, eds.), 2000, www.dcs.shef.ac.uk/ fabio/ecai-workshop.html.

[48] D Freitag and A K McCallum, *Information extraction with HMMs and shrinkage*, Proceedings of the AAAI-99 Workshop on Machine Learning for Informatino Extraction, 1999.

[49] Y Freund and R Schapire, *A short introduction to boosting*, Japanese Society for Artificial (1999).

[50] J Fukumoto, F Masui, M Shimohata, and M Sasaki, *Description of the oki system as used for muc-7*, 1998.

[51] R Gaizauskas and K Humphreys, *Xi: A simple prolog-based language for cross-classification and inheritance*, 7th International Conference on Artificial Intelligence, 1996.

[52] _____, *Using a semantic network for information extraction*, 1997.

[53] R Gaizauskas, T Wakao, K Humphreys, H Cunningham, and Y Wilks, *Description of the LaSIE system as used for MUC-6*, 1995.

[54] R Garigliano, A Urbanowicz, and D J Nettleton, *Description of the lolita system as used in muc-7*, 1998.

[55] B Gates and C Hemingway, *Business @ the speed of thought: Succeeding in the digital economy*, Warner Books, 2000.

[56] G Gay, A Sturgill, and W Martin, *Document-centered peer collaborations: An exploration of educational uses of networked communication technologies*, Computer Mediated Communication 4 (1999).

[57] Y Gil and V Ratnakar, *Trellis: An interactive tool for capturing information analysis and decision making*, 13th International Conference on Knowledge Engineering and Knowledge Management, Siguenza, Spain, October 1-4, 2002, 2002.

[58] _____, *Trusting information sources one citizen at a time*, First International Semantic Web Conference (ISWC), Sardinia, Italy, June 2002.

[59] H Glaser, H Alani, L Carr, S Chapman, F Ciravegna, A Dingli, N Gibbins, S Harris, M C Schraefel, and N R Shadbolt, *CS AKTive space: Building a semantic web application*, The Semantic Web: Research and Applications (First European Web Symposium, ESWS 2004) (2004), 417–432.

[60] T R Gruber, *A Translation Approach to Portable Ontology Specifications*, Knowledge Acquisition 6 (1993), no. 2, 199–221.

[61] _____, *Towards Principles for the Design of Ontologies Used for Knowledge Sharing*, Formal Ontology in Conceptual Analysis and Knowledge Representation (Deventer, The Netherlands) (N Guarino and R Poli, eds.), Kluwer Academic Publishers, 1993.

[62] A Gupta, B Ludascher, and M E Martone, *Knowledge-based integration of neuroscience data sources*, Statistical and Scientific Database Management, 2000, pp. 39–52.

[63] S Handschuh, S Staab, and F Ciravegna, *S-cream — semi-automatic creation of metadata*, 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), October 2002.

[64] _____, *S-CREAM - Semi-automatic CREAtion of Metadata*, Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02, Springer Verlag, 2002.

[65] S Handschuh, S Staab, and A Maedche, *CREAM — Creating relational metadata with a component-based, ontology driven framework*, In Proceedings of K-Cap 2001 (Victoria, BC, Canada), October 2001.

[66] S Harabagiu and S Maiorano, *Acquisition of linguistic patterns for knowledge-based information extraction*, LREC-2000, 2000.

[67] D Haussler, *Probably approximately correct learning*, American Association for Artificial Intelligence (AAAI90), 1990.

[68] J Heflin and J Hendler, *Searching the web with SHOE*, Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01, AAAI Press, 2000, pp. 35–40.

[69] J Henderson and E Brill, *Bagging and boosting a tree bank parser*, North America chapter of the Association of Computer Linguists (NAACL00), 2000.

[70] J Hendler, *Agents and the semantic web*, IEEE Intelligent Systems Journal **16** (2001), no. 2, 30–37.

[71] S Huffman, *Learning information extraction patterns from examples*, Springer, 1996.

[72] K Humphreys, R Gaizauskas, S Azzam, C Huyck, B Mitchell, H Cunningham, and Y Wilks, *University of sheffield: Description of the LaSIE-II system as used for MUC-7*, 1998.

[73] IEEE, *Special issue on information integration on the web*, IEEE Intelligent Systems, September/October 2003.

[74] J Kahan and M R Koivunen, *Annotea: an open RDF infrastructure for shared web annotations*, World Wide Web, 2001, pp. 623–632.

[75] M Kearns and U Vazirani, *An introduction to computational learning theory*, The MIT Press, 1994.

[76] C A Knoblock, S Minton, J L Ambite, P J Modi, N Ashish, I Muslea, A G Philpot, and S Tejada, *Modeling web sources for information integration*, Proc. Fifteenth National Conference on Artificial Intelligence, 1998.

[77] N Kushmerick, *Wrapper construction for information extraction*, Ph.D. thesis, University of Washington, 1997.

[78] N Kushmerick, D Weld, and R Doorenbos, *Wrapper induction for information extraction*, Internation Joint Conference of Artificial Intelligence (IJCAI-97), 1997.

[79] _____, *Wrapper induction for information extraction*, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997., 1997.

[80] J Liu, *Web intelligence (wi) : Some research challenge*, International Joint Conference on Artificial Intelligence (IJCAI'03) Invited Talk, Acapulco, Mexico, August, 2003.

[81] B Ludascher, A Gupta, and M Martone, *Model-based information integration in a neuroscience mediator system*, 2000.

[82] S Luke, L Spector, D Rager, and J Hendler, *Ontology-based Web Agents*, Proceedings of First International Conference on Autonomous Agents, LNCS. 1997.

[83] M Marcus, G Kim, M Marcinkiewicz, R MacIntyre, A Bies, M Ferguson, K Katz, and B Schasberger, *The penn treebank: Annotating predicate argument structure*, 1994.

[84] G A Miller, *Wordnet: An online lexical database*, 1990.

[85] S Miller, M Crystal, H Fox, L Ramshaw, R Schwartz, R Stone, and R Weischedel, *Bbn: Description of the sift system as used for MUC7*, Proceedings of the 7th Message Understanding Conference, 1998, *www.itl.nist.gov/iaui/894.02/related_projects/muc/*.

[86] T Mitchell, *Machine learning*, McGRAW-HILL, 1997.

[87] _____, *Extracting targeted data from the web*, Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (San Francisco, California), 2001.

[88] R Morgan, R Garigliano, P Callaghan, S Poria, M Smith, A Urbanowicz, R Collingham, M Costantino, C Cooper, and T Group, *Description of the lolita system as used in muc-6*, 1995.

[89] I Muslea, S Minton, and C Knoblock, *Wrapper induction for semistructured web-based information sources*, Proceedings of the Conference on Automated Learning and Discovery (CONALD), 1998., 1998.

[90] Z Nie and S Kambhampati, *Frequency-based coverage statistics mining for data integration*, 2003.

[91] Z Nie, S Kambhampati, U Nambiar, and S Vaddi, *Mining source coverage statistics for data integration*, Web Information and Data Management, 2001, pp. 1–8.

[92] J Quinlan and R Cameron-Jones, *Induction of logic programs: Foil and related systems*, New Generation Computing Jurnal **13** (1995).

[93] E Riloff, *Automatically constructing a dictionary for information extraction tasks*, National Conference on Artificial Intelligence, 1993, pp. 811–816.

[94] ———, *Automatically generating extraction patterns from untagged text*, AAAI/IAAI, Vol. 2, 1996, pp. 1044–1049.

[95] M Röscheisen, C Mogensen, and T Winograd, *Shared web annotations as a platform for third-party value-added information providers: Architecture, protocols, and usage examples*, Technical Report, Computer Science Department, Stanford University (1994).

[96] D Roth, *Learning to resolve natural language ambiguities: a unified approach*, Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence (Madison, US), AAAI Press, Menlo Park, US, 1998, pp. 806–813.

[97] D Roth and W Yih, *Relational learning via propositional algorithms: An information extraction case study*, IJCAI, 2001, pp. 1257–1263.

[98] ———, *Relational learning via propositional algorithms: An information extraction case study*, Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI), 2001, Seattle.

[99] M C Schraefel, N R Shadbolt, N Gibbins, H Glaser, and S Harris, *CS AKTive space: Representing computer science in the semantic web*, In Proceedings of World Wide Web Conference, 2004.

[100] N R Shadbolt, N Gibbins, H Glaser, S Harris, and M C Schraefel, *CS AKTive space or how we stopped worrying and learned to love the semantic web*, IEEE Intelligent Systems (2004).

[101] S Soderland, *Learning information extraction rules for semi-structured and free text*, Machine Learning **34** (1999).

[102] S Soderland, D Fisher, J Aseltine, and W Lehnert, *CRYSTAL: Inducing a conceptual dictionary*, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (San Francisco) (Chris Mellish, ed.), Morgan Kaufmann, 1995, pp. 1314–1319.

[103] S G Soderland, *Building a machine learning based text understanding system*, IJCAI-01 Workshop on Adaptive Text Extraction and Mining, Internation Joint Conference on Artificial Intelligence, 2001.

[104] C A Thompson, M E Califf, and R J Mooney, *Active learning for natural language parsing and information extraction*, Sixteenth International Machine Learning Conference (ICML-99), June 1999, pp. 406–414.

[105] Y Wilks, *Ie as a core language technology, what is IE?*, Information Extraction (M-T. Pazienza, ed.), Springer, 1997.

[106] Y Yao, N Zhong, J Liu, and S Ohsuga, *Web intelligence (wi): Research challenges and trends in the new information age*, Web Intelligence (WI-2001) Keynote Talk, Maebashi, Japan, October, 2001.