Acquiring Information Extraction Patterns from Unannotated Corpora

Neus Català i Roig

Memòria presentada al Departament de Llenguatges i Sistemes Informàtics de la Universitat Politècnica de Catalunya per a optar al títol de Doctora en Informàtica.

Sota la direcció de la Doctora Núria Castell Ariño

Barcelona. Juliol de 2003.

To my loved ones: Bruna, Júlia and Mario.

Acknowledgments

Carrying out this thesis required the support and help of a lot of people that, first of all, I would like to mention here.

I wish to thank my advisor Dr. Núria Castell, for her support and encouragement along these years.

I am grateful to my colleagues at the NLP Research Group of the Technical University of Catalonia and to the technical staff of the LSI department.

I would also like to thank to the anonymous reviewers of my PhD proposal for their constructive criticisms which helped me to improve this dissertation.

I cannot express my gratitude to Mario Martin. Without him none of this work could have done. He has encouraged me almost every day to follow my work, supported me, and taken care of me and our children.

Finally, I thank my parents and brothers that have given to me their support throughout the time spent in this work. Now they are aware that not only executives have no free time.

This research has been supported by several projects coming from diverse institutions: ITEM project number TIC96-1243-C03-02 and ALIADO project number TIC2002-04447-C02, funded by the Spanish government; EuroWordNet project number LE4003 and Meaning project number IST-2001-34460, funded by the European Union. Our group is recognized as a Quality Research Group (2001 SGR 00254) by DURSI, the Research Department of the Catalan Government.

Abstract

Information Extraction (IE) can be defined as the task of automatically extracting preespecified kind of information from a text document. The extracted information is encoded in the required format and then can be used, for example, for text summarization or as accurate index to retrieve new documents.

The main issue when building IE systems is how to obtain the knowledge needed to identify relevant information in a document. Today, IE systems are commonly based on extraction rules or IE patterns to represent the kind of information to be extracted. Most approaches to IE pattern acquisition require expert human intervention in many steps of the acquisition process. This dissertation presents a novel method for acquiring IE patterns, ESSENCE, that significantly reduces the need for human intervention. The method is based on ELA, a specifically designed learning algorithm for acquiring IE patterns from unannotated corpora.

The distinctive features of ESSENCE and ELA are that 1) they permit the automatic acquisition of IE patterns from unrestricted and untagged text representative of the domain, due to 2) their ability to identify regularities around semantically relevant concept-words for the IE task by 3) using non-domain-specific lexical knowledge tools such as WordNet and 4) restricting the human intervention to defining the task, and validating and typifying the set of IE patterns obtained.

Since ESSENCE does not require a corpus annotated with the type of information to be extracted and it does make use of a general purpose ontology and widely applied syntactic tools, it reduces the expert effort required to build an IE system and therefore also reduces the effort of porting the method to any domain.

In order to ESSENCE be validated we conducted a set of experiments to test the performance of the method. We used ESSENCE to generate IE patterns for

a MUC-like task. Nevertheless, the evaluation procedure for MUC competitions does not provide a sound evaluation of IE systems, especially of learning systems. For this reason, we conducted an exhaustive set of experiments to further test the abilities of ESSENCE. The results of these experiments indicate that the proposed method is able to learn effective IE patterns.

Resum

L'Extracció d'Informació (EI) és un terme que designa la tasca d'extraure de forma automàtica un tipus preespecificat d'informació continguda en un document de text. La informació extreta es tradueix al format requerit i és usada, per exemple, per generar un resum o com a índex per recuperar nous documents.

El problema principal que es planteja a l'hora de construir un sistema d'EI és com obtenir el coneixement necessari per identificar la informació rellevant en un document. La majoria de les aproximacions fetes fins ara fan ús de regles o patrons d'EI per representar el tipus d'informació que es vol extraure. Moltes d'aquestes aproximacions requereixen la intervenció de l'expert humà en diverses etapes del procés d'adquisició.

Aquesta tesi presenta un nou mètode per generar patrons d'EI, anomenat ESSENCE, que redueix de forma significativa la feina que ha de fer l'expert humà. Aquest mètode es basa en ELA un algorisme d'aprenentatge dissenyat específicament per adquirir patrons d'EI, que no fa ús de corpus anotats.

Els trets rellevants d'ESSENCE i ELA són: 1) permeten l'adquisició automàtica de patrons d'EI a partir de textos lliures representatius del domini, gràcies a 2) la seva habilitat per identificar regularitats al voltant de conceptes semànticament rellevants per la tasca d'extracció, 3) usant eines de coneixement lèxic que no són específiques del domini i 4) limitant la feina de l'expert a la definició de la tasca i a la validació i tipificació del conjunt de patrons d'EI obtinguts.

Atès que ESSENCE no requereix un corpus amb anotacions que assenyalin el tipus d'informació a extraure i que usa una ontologia i eines sintàctiques d'abast general, l'esforç de l'expert a l'hora de construir un sistema d'EI es redueix notablement i, per tant, també es redueix l'esforç necessari per usar el mètode sobre nous dominis.

Per tal de validar ESSENCE, hem realitzat una sèrie d'experiments que permeten obtenir una avaluació del mètode d'aprenentatge. Hem usat ESSENCE per generar patrons d'EI per a una tasca del MUC. No obstant, el mètode d'avaluació per les competicions del MUC no permeten una avaluació prou sòlida del sistemes d'EI, especialment pels sistemes que aprenen. Per aquest motiu, hem dissenyat un conjunt exhaustiu d'experiments per avaluar amb més profunditat les capacitats d'ESSENCE. Els resultats obtinguts per aquests experiments mostren que el mètode proposat és capaç d'aprendre patrons d'EI eficaços.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Information Extraction (IE) is a Natural Language Processing (NLP) task whose goal is to extract a previously specified type of information from a textual document. This task becomes specially important in our times when information on the Internet, the hugest source of information, is mostly available in the form of natural language texts. Unfortunately, although it is possible to identify and retrieve a large amount of documents relevant to a specific topic by using existing sophisticated search engines, it is no so easy to obtain what these documents are "saying" about the topic proposed. The former task is known as Information Retrieval (IR) while the second one is referred to as IE. An IE system should identify and extract relevant fragments of text within the document, without doing a deep comprehension of it. To decide whether or not a text fragment is relevant, one has to specify in advance the target information.

To illustrate the task of IE, an example concerning airplane crashes or accidents is shown in figure 1.1. The target information is the aircraft involved in the accident or crash, the location and the date of the crash, the number of victims in the crash, etc. The other facts appearing in these texts must be ignored. Results are usually presented to the user as a template filled with the extracted information, as shown in figure 1.2. Figures 1.3 and 1.4 show, respectively, a quite different example of document and filled output template for an IE task concerning lodging rentals.

The structure of the text documents from which the information must be extracted may vary depending on the source they are generated. From free text, such as newspaper or TV news, or even a research paper, where information is presented following non established order, to highly structured texts such as job advertisements publicly available on the Internet by following a predefined guideline. The text shown in 1.1 is an example of free text and the one shown in 1.3 is an example of structured text. The task of extracting information from

```
<DOC>
<DOCID> nyt960207.0722 </DOCID>
<TEXT>
SEATTLE - It's the phone call no one wants to get, but
everyone knows might come one day.
It came late Tuesday when Boeing got word that a chartered
757 aircraft crashed shortly after takeoff from the
Dominican Republic.  All 189 passengers are feared dead.
The crash, only the second in the history of the Boeing
757, came less than two months after an American Airlines
757 slammed into a mountain as it approached Cali,
Colombia.  Four people survived the Dec.  20 crash that
killed 160 people.  The cause has not yet been determined.
After hearing the news of Alas Nacionales Flight 301
Tuesday night, members of Boeing's Air Safety
Investigation Group monitored the situation throughout the
night and quickly assembled a team of safety experts to be
on standby in case they were needed at the crash scene.
One Boeing air safety investigator was expected to arrive
Thursday in Puerto Plata to assist a team from the
National Transportation Safety Board and the Dominican
Republic in trying to determine why the two-engine jet
crashed.  More Boeing engineers will be called in if
needed.
...
</TEXT>
</DOC>
```

Figure 1.1: A sample document from the aircraft crash domain.

one kind of document style and from the other one requires different linguistic resources and techniques. For structured text, information is located at the precise places intended to hold a particular kind of information. To extract the desired information from them it is enough to have rules close to regular expressions which are written by hand or automatically learned. For free text, the information to be found can be located wherever within the document and therefore more complex rules are required to detect and extract the target information. Commonly, complex rules for information extraction carry lexical, syntactic and semantic features, and linguistic experts are involved in the process of obtaining them.

Our method deals with the more complex document style, free text, because we are interested in exploring the possibilities of the application of NL advanced

```
Crash_Event1:
  Crash_Site:     Dominican Republic
  Crash_Date:     06-02-1996
  Aircraft:       757
  Airline:        Alas Nacionales
  Manufacturer:   Boeing
  Departure:      Dominican Republic
  Destination:    -

Crash_Event2:
  Crash_Site:     Cali (Colombia)
  Crash_Date:     20-12-1995
  Aircraft:       757
  Airline:        American Airlines
  Manufacturer:   Boeing
  Departure:      -
  Destination:    Cali (Colombia)
```

Figure 1.2: Filled event templates for the document in figure 1.1.

techniques (such as syntactic and semantic taggers or general-purpose ontologies exploitation), to automatically extract information from text by using extraction rules, then, without performing a full text understanding. This is contrasted with extraction from structured text because it doesn't require deep linguistic knowledge and can be solved by using techniques for learning grammars.

One way to introduce recent approaches to the IE task is by considering them in the context of the DARPA's Message Understanding Conferences (MUC). These conferences have motivated the research in IE and have also provided evaluation metrics in order to establish a quantitative evaluation for IE systems. A more detailed description of the tasks addressed by the MUCs is given in section 2.1.

Independently of the style of the document to be analyzed, a suitable system for IE should be able to fulfill the following features:

1. It must identify the target concepts located within the textual document being analyzed.

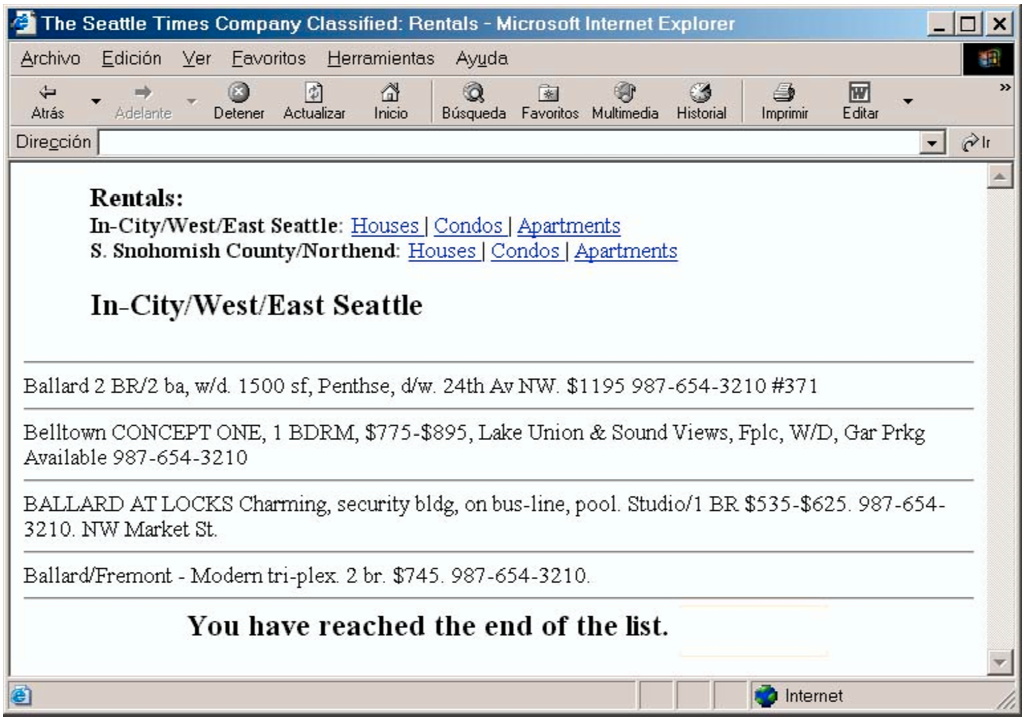


Figure 1.3: A sample rental advertisement from a web page.

2. It must establish the relationships between these concepts. Essentially, starting from the kind of facts to be extracted, the system must be able to find out which concepts are involved in a particular fact and the roles they play on it.

3. It must extract the target information and present it in the required format.

To build an IE system requires to design many and complex specific modules. The complete process of construction makes an extensive use of linguistic knowledge as well as specific abilities implemented mainly by NLP tools. Some of these tools are of general-purpose in the NL field such as sentence splitters, syntactic analyzers or semantic taggers.

But an IE system also requires domain specific components to identify domain specific relationships among relevant concepts in the text. Usually, domain specific components are customized for each domain involving several persons spending a lot of time. Moreover, the work done must be repeated for each new domain of extraction.

It seems clear that in order to build an IE system that could be easily adapted across different domains, tools that could also be easily adapted are required.

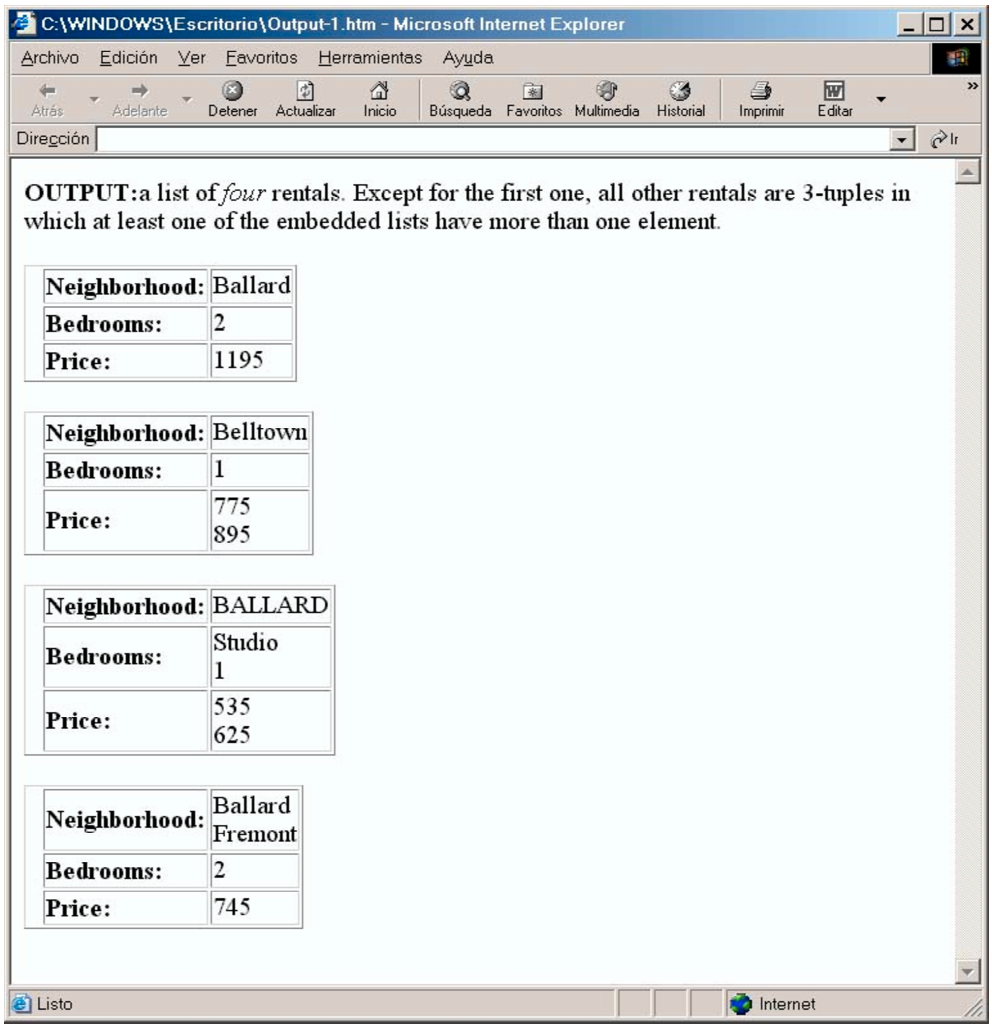


Figure 1.4: The filled output template corresponding to the document in figure 1.3.

In addition, if one wants that the building process has minimal human expert requirements then it would be a good deal to develop more efficient tools able to make the adaptation process as automatic as possible.

Recently, the successful application of corpus-based methods (also known as empirical methods) in computational linguistics to automate the acquisition of knowledge for NLP tasks (Cardie 1997) has boosted the research on learning methods for automatically build information extraction systems. Most of the empirical methods proposed for IE are corpus-based learning methods and they have been used for particular subtasks of IE, such as named entity identifica-

```
ID:: 1

Pattern:: * ( Digit )'BR' * '$' ( Number )

Output:: Rental {Bedrooms $1} {Price $2}
```

Figure 1.5: WHISK sample rule from rental advertisement domain. This rule looks for number of bedrooms and associated price.

tion, coreference resolution or extraction pattern acquisition. Each one of these methods is intended to improve the design of a particular component of the IE system.

Our work is mainly concerned with the design of one component of an IE system, concretely we focus on the acquisition of the less portable piece of the system: the set of information extraction patterns or rules. A common way of extracting the desired information is by using *IE patterns* (also known in the literature as *extraction rules* or *conceptual patterns*). An IE pattern consists of a set of constraints that a fragment of text must satisfy in order to extract relevant information from it, together with an indication of what information of that text should be extracted. For example, if we are looking for reports of aircraft crash locations, we might use a pattern of the form <AIRCRAFT crashed in LOCATION>, where aircraft and location are semantic tags for noun groups. This pattern is satisfied by sentences containing a noun group semantically tagged as AIRCRAFT followed by the word "crashed", the preposition "in" and a noun group semantically tagged as LOCATION.

The kind and amount of information an IE pattern should hold mainly depends on the task itself and on the kind of textual documents to be processed. For structured or semi-structured texts, extraction patterns are close to regular expressions. Figure 1.5 shows an example of an extraction rule produced by WHISK (Soderland 1999) system applied to rental advertisements in HTML texts. WHISK's rules have two components: the *pattern* part, that identifies the possible slot fillers, and the *output* part, that specifies the exact pieces of the phrase to be extracted. If the entire pattern matches, slots of the output template are filled according to the output part of the rule. Parenthesis indicate a phrase to be extracted. The phrase within the first set of parenthesis is bound to the variable $1 in the output format, the second to the variable $2 and so on.

For free texts, extraction patterns usually carry both syntactic and semantic knowledge to represent how the relevant information is expressed in text. Figure 1.6 shows another example of an extraction pattern (also called concept node) in the domain of natural disasters, generated by AUTOSLOG (Riloff 1993). This

```
Concept-Node Definition:

   Concept = Damage
   Trigger = "destroyed"
   Position = direct-object
   Constraints = ((physical-object))
   Enabling Conditions = ((active-voice))
```

Figure 1.6: Concept node definition for extracting “damage” information.

rule is triggered by sentences containing the verb “destroyed” in the active voice and look for the semantic class *physical-object* in the direct object.

Methods for acquiring IE patterns for a task must take into account the time cost of manual intervention, usually done by an expert. Obviously, patterns acquired for one domain can rarely be reused for a new domain. Therefore, these methods should allow:

1. The acquisition of IE patterns for any domain insofar as this is possible, in order to allow the system to be ported across domains, and

2. The expert intervention to be minimized.

Expert intervention can be required in different ways. For example, in systems that learn IE patterns from an annotated corpora, it is necessary to have an expert of the domain and the task to suitably annotate the texts for the target information. In other approaches, the IE patterns are obtained by interaction with the expert during the processing of a text. Chapter 2 will provide a more detailed description related to the work to be done by the expert on the different reviewed approaches.

The method we propose aims at notably reducing the expert intervention in the process of generating IE patterns for a task or domain of extraction. This is one of the main goals of this thesis that next section will present.

1.1 Goals of the Thesis

This work is concerned with building of IE systems for non-structured text. More specifically, it focuses on the acquisition of extraction patterns for information extraction.

IE pattern acquisition, as already mentioned, is one of the hardest tasks in the process of building an IE system and often requires the human expert intervention. Besides, several linguistic knowledge resources are needed during the

acquisition process, and some of them also involve an expert to tune them for the specific task or domain. The high cost of having available an expert and the large time spent by the expert to perform these tasks make the building of IE systems very expensive and becomes a bottleneck of the whole process.

In short, the goal of this thesis is to reduce as much as possible the intervention of the expert to alleviate the task of acquiring extraction patterns for an IE system. We do this by means of a new method, ESSENCE, that introduces Machine Learning (ML) techniques which allow the expert to concentrate on defining the task and validating results. On the other hand, the method makes use of available linguistic knowledge resources which are general enough to require minimal tuning overhead.

In order to reach this generic goal, this thesis pursues the solving of a set of issues that are described in the following sections.

Reducing Expert Intervention in Acquiring IE Patterns

Since the core of an IE system is its extraction pattern database, research in IE has been mainly focused on the issue of acquiring an adequate set of extraction patterns for a given IE task.

Approaches to the acquisition of extraction patterns differ in diverse properties. Among these properties there is the amount and kind of expert intervention required. The knowledge engineering (handcrafting) approach is based on a knowledge engineer who is in charge of writing the extraction patterns through examination of a corpus and who does the tuning of the patterns after testing them on the corpus. This approach is tedious and time consuming; moreover, it becomes subjective because results depend on the human expert rather than on actual texts.

In recent years, attempts have been made in some IE systems to use ML techniques in order to semi-automate the pattern acquisition process. This allows the expert to be alleviated of handcrafting IE patterns and therefore it reduces the expert intervention. Nevertheless, the IE pattern acquisition process still remains time consuming because most ML techniques applied are usually supervised, that is, are based on learning from a set of examples that must be selected and annotated by a domain expert. As examples we mean instances of the concept being learned, whether in the form of manual annotations in the training text or in the form of filled templates. To provide the system with a set of examples is a time-consuming task that produces a bottleneck in the process of acquiring IE patterns.

One of the issues this thesis addresses is the acquisition of a set of IE patterns from unannotated text, therefore without user-labelled examples indicating

where the information to be extracted can be found in texts. To achieve this goal we propose a learning algorithm which relays on:

1. A simple set of trigger keywords (provided by the user and intended to focus on candidate sentences for carrying relevant information),

2. A set of we call extracting synsets (semantic description of the target information), and

3. The hypothesis that the ways to express a particular event are limited in number, are frequently used in texts of the domain, and present a similar structure: information of the event is surrounding trigger keywords.

Minimizing Handcrafted Linguistic Resources

To develop the different components of an IE system, several linguistic resources and tools are needed. Among them we can find lexicons or ontologies, that is, repositories of lexical knowledge.

Given that the IE task is domain specific by nature, many IE systems are provided with lexical resources that have been manually prepared for the intended purpose. For example, CRYSTAL (Soderland et al. 1995a) and PALKA (Kim and Moldovan 1995) systems require both a lexicon that maps each word to its semantic class information and a domain specific semantic class hierarchy. To handcraft this kind of knowledge is: a time-consuming process, also a subjective task and, in most cases, has to be done every time the system is moved to a new domain. In order to ease the process of building IE system components it would be desirable to minimize the manual customization of the needed lexical resources.

One way to have lexical information is by using available resources of lexical knowledge. Our method makes use of an available general purpose lexical database: WordNet[1] (Miller 1995). We make an extensive use of WordNet, especially for lexical processing and pattern generalization. WordNet has been successfully used in other NLP tasks, such as word sense disambiguation (see (Agirre and Rigau 1996), (McCarthy 1997) and (Mihalcea and Moldovan 1998)), information retrieval (see (Gonzalo et al. 1998)) and text categorization (see (Scott and Matwin 1998)), but very few attempts have been made of using it in IE systems (Chai 1998). Moreover, some reports, for instance (Grishman et al. 1992), have criticized the use of WordNet in IE mainly due to: (1) word sense ambiguity, (2) irregular granularity of description in different branches of the concept hierarchy (and thus irregular depth of branches in the hierarchy), and (3)

[1] http://www.cogsci.princeton.edu/∼wn

lack of covering of domain specific vocabulary. Our experiments using WordNet prove that only very specific vocabulary, such as company names or names of small villages, was not covered by WordNet. Most of the missing vocabulary can be supplied to the system by means of gazetteers or domain specific word lists.

The customization of lexical tools then is confined to define domain specific vocabulary not covered by general tools. It is worth mentioning that some approaches have been proposed to automatically acquire domain specific knowledge (Riloff and Jones 1999). The technique presented in this work can also be used to generate domain specific lexicons and preliminary experiments carried out on this task showed promising results. May be we could use this technique to finally avoid the need for handcrafted linguistic resources.

Learning from a Small Set of Documents Without Annotations

As mentioned above, most ML techniques applied to extraction pattern acquisition are usually supervised, therefore, they rely either on a training corpus that has been suitably annotated for the intended IE task or on an unannotated training corpus along with the correct answers to be produced. For example, MUC systems were provided with a set of documents (free text) and the set of filled output templates that should be produced for each document. But, in general, there are no corpora annotated with domain specific labels. Thus, for each application domain a new annotated corpus must be created.

This problem is alleviated by recent semi-supervised learning algorithms (see section 2.3.2) because they do not need an annotated corpus, but then to learn IE patterns achieving good recall scores, they require a training corpus containing a large number of documents (of the order of thousands of texts). This requirement reduces to some extent the benefits of learning without annotated texts because it increases the computational resources needed to learn and, even worse, it increases the effort of the expert in obtaining the training texts, and also in monitorizing the learning process and validating results. In order to reduce the work to be made by the expert we should reduce also the number of texts needed for learning. Another goal of this thesis is to allow learning from unannotated corpora but from a relative small set of documents (of the order of hundreds of texts).

Summary of goals

We can summarize in one sentence the main goals of this thesis as:

> **Goal:** To reduce the human expert intervention in the process of acquiring Information Extraction patterns by learning from an *unannotated domain corpus* composed of *few texts* and by *reducing as much as possible the amount of linguistic resources to be handcrafted*.

This dissertation presents a new method, called ESSENCE, which is able to learn IE patterns without a set of annotated examples. ESSENCE requires a corpus of the domain, a semantic description of the target information, and a set of trigger words for this target information (called *context keywords* in our method) that are proposed by a domain expert. From this knowledge, the method automatically obtains specific patterns which are then generalized to more general patterns that could be applied to new texts. The output is a set of patterns that cover sentences containing possible relevant information to be extracted. Finally, the expert should check the set of patterns and select those useful for extracting the target information. The expert's effort is reduced from reading and tagging documents in the training corpus to accepting/rejecting relatively few patterns.

The primary focus of this thesis is on the automatic acquisition of IE patterns. We do not deal with the other issues involved in a complete IE system, such as coreference resolution or template merging.

1.2 Roadmap

The remainder of the dissertation is organized as follows. Chapter 2 presents background knowledge on information extraction and reviews some approaches for automatically generating IE patterns using ML techniques. Since the main goal of the thesis is to learn IE patterns from free texts without user-labeled examples, the reviewed approaches are divided into supervised learning approaches and semi-supervised learning approaches to IE pattern acquisition from free texts.

The ESSENCE method is presented in chapter 3. The motivations and main distinctive features of the new method are summarized in the beginning of the chapter. This is followed by sections that describe in detail each component step of the method.

Chapter 4 describes the ESSENCE Learning Algorithm (ELA) which is the core of the ESSENCE method. After a brief discussion of unsupervised learning

methods, the generalization procedure used in the algorithm is presented and justified. The last section of the chapter describes the details of the learning algorithm proposed.

Chapter 5 shows a case application of the ESSENCE method to a MUC-style task. The chapter includes a detailed description of the steps followed by the expert to apply the method. Then, it reports results of testing ESSENCE on a MUC extraction task.

In order to complete the evaluation of the proposed method, chapter 6 reports the results of an exhaustive set of experiments conducted to test the performance of ESSENCE with different number of training texts and different values for the parameters of the learning method.

Finally, chapter 7 gives a discussion of some features of ESSENCE and suggests some ways to further improve it. This chapter concludes with a brief summary of the thesis and reviews its main contributions.

Chapter 2

Background

An IE system performs two primary tasks. First, the system locates the relevant information in the text of a document by means of a text analysis. Essentially, the system identifies the entities of interest and the relations between such entities (as events and their participant entities) that meet the specifications for the IE task. Second, the system extracts the target information and presents it in the required format.

The task of locating the relevant information in a text is usually carried out through a set of sequential phases. First step performs a lexical analysis of the input text that includes processes such as morphological analysis and part-of-speech tagging. This step is completed by a named entity recognizer which is in charge of identifying and categorizing particular expressions of entities, such as proper names, dates or quantities. After lexical analysis, a parsing process is done. Some systems do a full parsing to obtain a complete syntactic structure of each sentence; other systems use a partial parser to identify major syntactic components of each sentence, such as noun groups, verb groups and prepositional groups. During semantic tagging, not performed by all IE systems, the system attaches semantic labels to the headword of each syntactic component. Following all these processes, the system identifies relevant entities and specific relationships between them in the text as target information. This is usually done by applying to the text a set of patterns which represent how relevant information is typically found in texts of the domain.

In the second task, the system processes the entities and relations identified as target information to generate the output as required. Prior to output generation, the system resolves coreferences, combines disperse information relative to an event into a single event structure and, if it is necessary, makes inferences to derive information not explicitly available in the text.

The general structure of an IE system is depicted in figure 2.1. However,

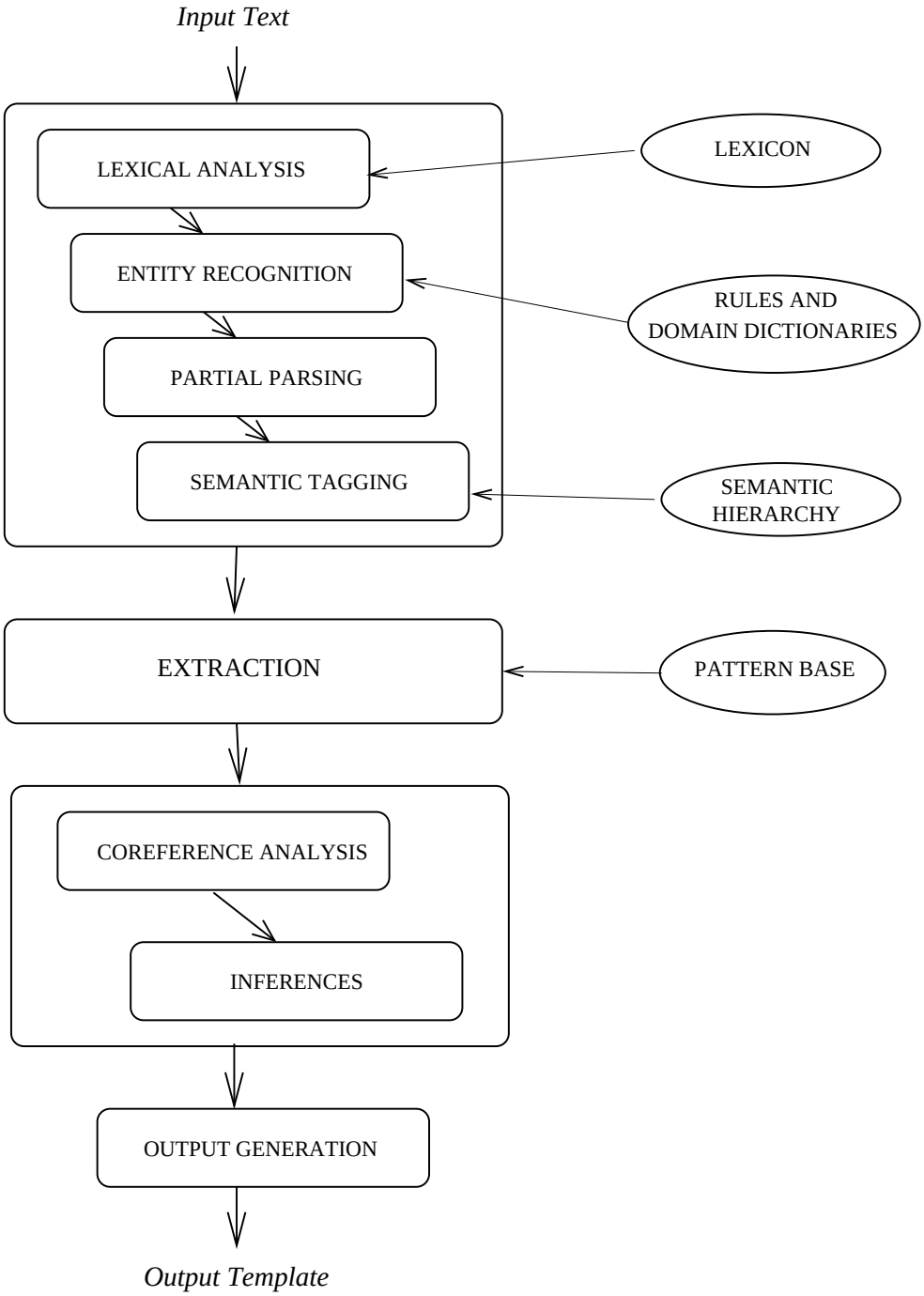


Figure 2.1: General architecture of an IE system.

depending on the requirements of a particular extraction task, it might be necessary to add some additional processes and/or knowledge resources to the general architecture. For example, dealing with a specific domain such as Hospital Discharge domain requires to analyze unstructured texts (usually containing abbreviations and incomplete sentences) and requires also knowledge about semantic information relative to specific medical concepts.

The first part of this chapter gives an introduction to the notion of IE as promoted by the Message Understanding Conferences (MUCs) and to the performance evaluation of IE systems. The MUC programme finished in 1998 but the IE technology is an on-going research field supported by new programmes such as the Linguistic Research and Engineering program[1] of the European Union or the TIDES(Translingual Information Detection, Extraction and Summarization) program[2] with the support of the DARPA agency. We will shortly review some recent and actual IE projects within these programmes. The last part of this chapter is centered on the issue this thesis addresses, that is, on the acquisition of extraction patterns. It reviews related approaches in automatic IE pattern acquisition.

2.1 The Message Understanding Conferences

The series of Message Understanding Conferences (MUCs[3]) started in 1987 and a total of seven MUCs have been taken place to 1998. They have been organized by the Naval Research and Development Group (NRAD) with the support of the Defense Advanced Research Projects Agency (DARPA[4]). The MUCs form the IE component of the TIPSTER[5] Text Program (Grishman 1996). The TIPSTER initiative has also focused on document detection (text retrieval and message routing) promoted by the TREC (Text REtrieval Conference) competitions, and on text summarization.

The main objective of the MUC conferences has been to promote research in IE and to provide IE systems with a quantitative evaluation method. A MUC can be seen as a competition where participant groups develop a system to process a set of texts and extract from them the type of information required. In a first stage, each participating group develops a system able to extract from a set of sample texts the particular information specified for the task. In a second stage, each system (the same as the first stage) is applied to a new set of texts and is

[1] http://www.ejeisa.com/nectar/t-book/html/lre.htm
[2] http://www.darpa.mil/iao/TIDES.htm
[3] http://www.itl.nist.gov/iaui/894.02/related_projects/muc
[4] http://www.darpa.mil
[5] http://www.ldc.upenn.edu/Catalog/Tipster.html

evaluated by comparing the outputs of the system with the correct answers (*answer keys*) produced by hand. Each conference focuses on a particular *domain* and *scenario*. The domain stands for a class of texts, for example, patient's medical reports. The scenario stands for the set of facts to be extracted, for example, symptoms and diagnoses. The target information consists of a previously specified set of entities and their attributes, together with their relationships and events relating to them. This information is usually represented in the form of *templates* whose slots must be filled.

Following we present a brief description of the MUC evaluations in chronological order. MUC-6 and MUC-7 are explained to a large extent in order to introduce some concepts that will be referred to throughout this document.

MUC-1 and MUC-2 were held in 1987 and 1989, respectively, and both shared the domain of tactical naval operations reports on ship sightings and engagements. In the second one, a particular task was specified and there were a template and rules for filling the slots. There was too a set of answer keys and evaluation criteria.

In MUC-3 (MUC 1991) and MUC-4 (MUC 1992), the domain was news wire stories about terrorist attacks in Central and South American countries. For the MUC-4, the organization made some changes to the evaluation method. To guarantee system independence from the training data, participating systems were given a set of training texts with the associated templates and were evaluated by running the system on a different test set with new associated templates.

For MUC-5 (MUC 1993) two domains were tested, international joint ventures and microelectronics products announcements, in two languages, English and Japanese. The complexity of the tasks grew up and new evaluation metrics were used.

The domain of the scenario for MUC-6 (MUC 1995) was management succession events in financial news stories. There were a menu of four tasks from which each participant could choose their preferences. The four tasks were: Named Entity (NE) recognition, Coreference (CO) resolution, Template Element (TE) filling and Scenario Template (ST) filling. These tasks were proposed in order to address the main goals of the MUC-6 and future conferences:

- To identify domain-independent components from the whole set of components being developed for building IE systems. Domain-independent components would be of practical use and could be developed automatically. To meet this goal, the organization developed the Named Entity task, which involved identify and categorize particular expressions of entities (organizations, persons, locations), times (dates, times), and quantities (monetary values, percentages) by inserting SGML tags into the text.

- To focus on portability in the IE task, that is, on the ability to rapidly tune a system to extract information about a different class of events. To increase portability, the organization proposed to standardize low-level objects (objects that could represent participants in an event such as people, products, etc.). These objects can be seen as basic classes that could be involved in different types of events. They were named Template Elements. The detection of specific relations holding between template elements relevant to the target information is the task of Scenario Template filling.

- To encourage work on deeper understanding. This was an attempt to move the trend of shallow understanding techniques performed by participating systems towards deeper understanding mechanisms. The three tasks to meet this goal were: Coreference (the task of inserting SGML tags into the text to link strings that represent coreferring noun phrases), Word sense disambiguation and Predicate-argument structure. At the end, the last two tasks were not considered.

MUC-7 (MUC 1998) was the last in the series of MUC evaluations. The main motivations of MUC-6 remained but more emphasis was placed on the minimal use of non-NLP requirements and on portability. There was a clear division between domain-independent information extraction tasks and domain-dependent information extraction tasks. On one hand, the extraction of named entities (NE and CO) and their attributes (TE) as well as facts about those entities (TR[6]) are considered as domain-independent tasks because they can be extracted from texts without knowing what role those elements are playing on. On the other hand, the extraction of information about events which involve the extraction of only particular entities that participate in the events (ST), is considered as a domain-dependent task because the extraction can be only performed from relevant texts and obeying the specific guidelines given for the task definition.

In MUC-7, named entities were defined as proper names (person, organization and location names) and quantities of interest (dates, times, percentages, and monetary amounts). Entities were marked within the text stream by using SGML annotations[7]. The task of coreference only dealt with the identity type[8]. For Template Elements four attributes or slots were required: name (which included all aliases), type (one of person, organization, artifact or location), descriptor (all substantial descriptors used in the text, such as “capt.” or “the ABC

[6]Template Relation (TR) are general relational objects which involve TE objects, for example, organizations and their locations or persons and the organizations they are employed.

[7]The Named Entity task was carried out in English, Chinese and Japanese.

[8]Task definition for coreference covered only “identity” relations for noun phrases; it did not include coreference among clauses nor other kinds of coreference relations, such as part/whole.

Corp. unit") and category (which depends on the element involved, for example, persons can be civilian, military or other; artifacts were limited to vehicles that can be for travelling on the ground, through water or by air; etc.). Template Relations were limited to relationships with organizations: employee_of, product_of and location_of. At the end, each Scenario Template captured information about a particular event in which objects (relational and low-level objects) participated.

The domain for training (dry-run evaluation) was airline crashes and the domain for testing (formal-run evaluation) was launch events. We used the dry-run texts to perform our experiments and we used the available answer keys to make the evaluation. In this thesis, we focus on the acquisition of patterns that extract information for the domain-dependent task (Scenario Template). The ST task definition makes reference to the Template Element and Template Relation objects obtained through the domain-independent tasks (NE, CO, TE and TR). In our work, we do not explicitly perform these tasks. We prove that to find out the required information about a particular event there is no need of having to find all entities in the text but just those entities that participate in the particular event.

2.1.1 Evaluating MUC-like IE Systems

The MUC evaluation is performed by means of a scoring software that compares the output of a system (responses) with the output of human linguists (answer keys).

The input of the system being evaluated is a set of documents containing text from news stories or some other natural language source. The system analyzes the text and produces a set of objects. The structure of the objects depends on the task.

The scorer receives the responses file and the keys file, and it aligns objects in one file with objects in the other file. In general[9], the scores are calculated by counting how many fills agree for each aligned object.

The metrics used for evaluating MUC systems have been evolved since the first evaluation. These metrics are similar to those used for IR: Recall (R) and Precision (P). For IE, recall is redefined as the fraction of required information that has been correctly extracted and precision is redefined as the fraction of the extracted information that has been correctly extracted. These metrics have been redefined for the IE task to allow for some new situations that don't happen in IR such as overgeneration (Lehnert and Sundheim 1991).

Recall and precision evaluate system performance at the level of slots, because they measure slots filled in the templates that the system is able to find

[9]Scores for coreference are based on the agreement of equivalence classes.

in text. Since MUC-3, another metric was also defined at the level of texts, the text-filtering metric, to measure the ability of a system at separating documents into relevant/non-relevant categories.

By definition, recall and precision are negatively correlated, i.e. pushing recall causes a fall in precision. The van Rijsbergen's F-measure (F), as used in IR, provides a method for combining recall and precision into a single measure. As can be seen in the following formula:

$$F = \frac{(\beta^2 + 1.0) * P * R}{\beta^2 * P + R} \tag{2.1}$$

recall and precision have relative weights. The β parameter determines the relative importance given to recall over precision. If recall and precision are equally important, β is set to 1.0.

We make use of the same measures to evaluate the method we present in this thesis. The set of patterns obtained by our method is able to extract information for some of the slots of the output templates for the ST task, but not all of them. The official scorer can't deal with partial results and therefore we manually calculate the values for the measures. For this reason, we can't perform a fair comparison between our work with the MUC-7 participating systems.

2.2 Some Recent and Actual IE Projects

The Message Understanding Conferences and the TIPSTER program have boosted research in IE. In addition to MUC evaluations several projects have been or are being developed within the IE field. Some of these projects are applications of the IE technology to specific tasks while others are focused on developing tools for IE in various European languages. This section doesn't give an exhaustive historical review of IE projects, but that makes a survey of recent and actual IE projects which do research to address the specific problems this work is concerned with.

Under the Language Engineering (LE) program of the European Union, several IE projects are being or have been supported. The TREE[10] (TRans European Employment) project will provide a multilingual[11] World-Wide Web employment service for job seekers to view details of professional opportunities in their own language. The AVENTINUS[12] (Advanced Information System for Multi-

[10] http://www.mari.co.uk/tree/

[11] The initial prototype system currently implemented can store and retrieve job advertisements in three languages, English, Flemish, and French, with a small Swedish retrieval module.

[12] http://www.dcs.shef.ac.uk/research/groups/nlp/funded/aventinus.html

national Drug Enforcement) project built a system to seek information on drug enforcement distributed over different international sources; the system supports multilingual search of text bases other than in the user's native language. The ECRAN[13] (Extraction of Content: Research at Near-Market) project aims at extracting information from full-text on the basis of POS taggers, robust parsers and its major concern is on automatic lexical acquisition and automatic lexicon tuning; the system's users can access information sources in a range of European languages (English, French, Italian). The SPARKLE[14](Shallow PARsing and Knowledge extraction for Language Engineering) project developed shallow parsing technology in four European languages together with corpus-based lexical acquisition techniques, and applied parsers to multilingual information retrieval and speech dialogue systems. The Proteus[15] project conducted a wide range of research related to IE including name extraction, event extraction and application of unsupervised methods, in several languages (English, Japanese, Spanish and Chinese). The main goal of the FACILE[16](Fast Accurate Categorization of Information using Language Engineering) project was to build a system capable of categorizing financial text to high degree of detail.

The GENIA[17] project seeks to automatically extract useful information from texts written by scientists to help overcome the problems caused by information overload. Currently, they are working on the key task of extracting event information about protein interactions in the micro-biology domain, but the basic methods should be generalizable to knowledge acquisition in other scientific and engineering domains.

Under the Translingual Information Detection, Extraction and Summarization (TIDES) program, supported by the DARPA agency, several projects are collaborating to develop resources and tools for various language engineering tasks. The tasks include Information Extraction, Machine Translation, Summarisation and Cross-Language Information Retrieval.

In this thesis we present a new method for extracting information from free text, also called unstructured text. The projects just mentioned refer to systems dealing with free text, but there are also systems that apply IE techniques to process rigidly structured text (mainly coming from the Internet) named *wrappers*. The problem of extracting information from structured text is somewhat different from ours in that it works on different domains and it makes a very limited use of linguistic knowledge. Some examples of wrapper induction systems are

[13]http://www.dcs.shef.ac.uk/research/ilash/Ecran/overview.html
[14]http://www2.echo.lu/langeng/en/le1/sparkle/sparkle.html
[15]http://nlp.cs.nyu.edu/
[16]http://www.quinary.com/pagine/innovation/facproj.htm
[17]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/

(Kushmerick 1997; Kushmerick 1998), (Muslea et al. 1998a) and (Hsu 1998).

2.3 Related Work in IE Pattern Acquisition

The process of acquiring IE patterns is one of the most difficult steps when building an IE system. Researchers have tried to facilitate this process in different ways, for example, by developing tools to assist the expert in writing patterns or by designing ML tools to automatically, or semiautomatically, acquire patterns from training corpora. The subject of this thesis is centered on the latter approach, in particular on the automatic acquisition of IE patterns from free text. In this section we review some related efforts for learning IE patterns used for free text. A summary of work on learning methods for IE pattern acquisition can be found in (Yangarber and Grishman 2000). In (Muslea 1999) there is a review of the different types of extraction patterns that are generated by ML algorithms. A detailed survey on the use of empirical ML methods for the entire IE task, and in particular for the pattern acquisition component of an IE system, can be found in (Cardie 1997).

The difference between corpus-based methods for learning IE patterns is made up of many and varied factors such as the type of patterns being learned, the kind and amount of training corpus required, the learning strategy or the degree of expert intervention in the acquisition process. Our primary criterion for classifying ML approaches to IE pattern acquisition is the degree of expert intervention. From this point of view we can notice how IE systems tend to reduce expert intervention in favor of using unsupervised techniques.

2.3.1 Supervised Machine Learning Approaches

Most of the first IE systems that tried to automatize the process of acquiring patterns often required the assistance of an expert. The kind and degree of the expert's assistance varies from one system to another one. For example, the expert can be required for annotating the training corpus, for manually reviewing the learned patterns or to online assist the learning process.

AutoSlog (Riloff 1993) was one of the first systems to take advantage of ML techniques for obtaining IE patterns. AutoSlog learns patterns (called *concept node* definitions) from an annotated corpus (training corpus) in which the information to be extracted is marked and tagged with IE-task specific semantic tags. The process consists of specializing a set of general syntactic linguistic patterns by searching in the training texts for positive examples that can instantiate them. The IE patterns learned consist of a trigger word, a syntactic context and the

```
Concept-Node Definition:

   Name = target-subject-passive-verb-bombed
   Trigger = bombed
   Variable Slots = (target (*S* 1))
   Constraints = (class phys-target *S*)
   Constant Slots = (type bombing)
   Enabling Conditions = ((passive))
```

Figure 2.2: Concept node definition for extracting the "target" of a terrorist attack.

information to be extracted. A concept node is activated when the trigger word appears in the linguistic context, and it is able to infer the role played by the target information in this context. A concept node is able to extract only one single slot per sentence. Figure 2.2 shows an example of a concept node definition. The linguistic pattern that applies in this example is <subject> passiveverb; this pattern says that the subject must be a physical target ("Constraints"), and the enabling conditions require a passive construction. The triggering word is "bombed" and the variable slots specify that the information to be extracted is the subject of the sentence (*S*).

AutoSlog uses a one-shot learning algorithm designed specifically for the IE task. In addition to the annotated training corpus, it requires a small lexicon with semantic information, a partial parser which is able to assign the semantic information to nouns and modifiers, and a predefined set of linguistic patterns (approx. 13) that are mostly domain independent.

In this approach, a human expert must examine the set of patterns learned in order to decide which ones to keep.

CRYSTAL (Soderland et al. 1995a) is another system that inductively generates a dictionary of *concept definitions* that cover the positive examples contained in the training texts.

CRYSTAL's concept definitions are multi-slot rules. They allow both semantic and exact word constraints on any component phrase. Figure 2.3 shows an example of a concept definition that applies to the sentence "He succeeds Jack Harper, a company founder who was named chairman". From this sentence, CRYSTAL creates a case frame with Person_In slot filled by "Jack Harper, a company founder", and the Position slot filled by "who was named chairman". A later processing will eliminate nonessential words included as part of the extracted information.

The learning process is an *iterated bottom-up covering algorithm* which ob-

```
Concept type: Succession Event
Constraints:
   OBJ::
     Classes include: <Person Name>
     Extract:         Person_In
   REL-OBJ::
     Terms include:   WHO NAMED
     Classes include: <Corporate Post>
     Extract:         Position
```

Figure 2.3: A CRYSTAL's concept definition. It looks for *Person_In* in the direct object and *Position* in a relative clause. The relative clause must also include the terms "who" and "named".

tains a new IE pattern, that covers some positive examples and does not cover negative examples, at each iteration. Learning finishes when all positive examples are covered. In addition to the training texts, CRYSTAL also makes use of a partial parser, a domain specific semantic hierarchy and associated lexicon created for the task by the expert.

The LIEP (Huffman 1995) system induces extraction patterns from positive instances provided by an expert and events to be extracted from them. The induced patterns are multi-slot rules which are able to identify semantic relationships between two target noun phrases (role-filler constituents) in a sentence. LIEP's patterns cannot handle single slot extraction.

A LIEP pattern consists of both syntactic constraints between constituents (e.g., constituent A is the subject of a verb and constituent B is the direct object of the same verb) and semantic constraints (e.g., the head of the constituent A is a "personname"). The LIEP pattern from figure 2.4 extracts the name of a person, the title position in a company to which the person is being moved and the name of the company. The syntactic constraints indicate that PNG is the subject of a sentence that also contains a verb group followed by a prepositional phrase; the semantic constraints specify that PNG is a person name, TNG is a title position, CNG is a company name, the verb is one of "named", "elected" or "appointed" and it used in its passive form, and the preposition for the prepositional phrase is one of "of", "at" or "by".

As stated in (Huffman 1995), LIEP's learning can be seen as an example of Explanation Based Learning (EBL) with an overgeneral and incomplete domain theory. LIEP's patterns are induced from positive examples only; these examples are provided by an expert who interactively identify events (entities of interest

```
n_was_named_t_by_c:
  noun-group(PNG,head(isa(person-name))),
  noun-group(TNG,head(isa(title))),
  noun-group(CNG,head(isa(company-name))),
  verb-group(VG,type(passive),
             head(named or elected or appointed)),
  preposition(PREP,head(of or at or by)),

  subject(PNG,VG),
  object(VG,TNG),
  post_nominal_prep(TNG,PREP),
  prep_object(PREP,CNG)
==> management_appointment(M,person(PNG),
                           title(TNG),company(CNG))
```

Figure 2.4: A LIEP information extraction pattern. It identifies persons moving into a company management position.

and combinations of them that signify events to be extracted) in texts.

PALKA (Kim and Moldovan 1995) inductively builds phrasal patterns that define a sequence of lexical items or semantic categories in the sentence in order to extract information. PALKA's patterns are similar in form to AutoSlog's concept nodes. They are represented as a pair of a meaning frame defining the types of information to be extracted, and a phrasal pattern describing the syntactic ordering. To combine a meaning frame and a phrasal pattern, each slot of the frame is linked[18] to the corresponding element in the phrasal pattern. This representation is called the *FP-structure* (Frame-Phrasal pattern structure). By matching the phrasal pattern to a sentence in the input text, the FP-structure is activated and then the meaning frame is used to extract the relevant information from the sentence. Figure 2.5 shows an example of an FP-structure able to extract information concerning a terrorist incident. By matching this pattern to the sentence "At 0115 this morning (0415 GMT) incendiary bombs were hurled at a Mormon temple at Nunoa district, in Santiago.", it will produce a bombing incident type with instrument "incendiary bombs" and target "Mormon temple".

PALKA performs both generalization, to include a new positive instance, and specialization, to avoid a negative instance, of an initial pattern.

This method requires more background knowledge than the previous ones: a set of annotated training texts, a set of predefined keywords used to trigger patterns, a concept hierarchy, a domain specific semantic hierarchy and an as-

[18]These mappings are acquired from filled output templates.

```
FP-structure = Meaning Frame + Phrasal Pattern

Meaning Frame: (BOMBING
                  agent: ANIMATE
                  target: PHYSICAL-OBJ
                  instrument: PHYSICAL-OBJ
                  effect: STATE)

Phrasal Pattern: ((BOMB) BE HURT AT (PHYSICAL-OBJ))

FP-structure:(BOMBING
                target: PHYSICAL-OBJ
                instrument: BOMB
                pattern: ((instrument) BE HURT AT (target)))
```

Figure 2.5: The frame-phrasal pattern representation used by PALKA. This FP-structure is able to extract the "instrument" and the "target" objects concerning a bombing incident.

sociated lexicon. The concept hierarchy contains generic concept definitions for each kind of information to be extracted.

A different method for learning IE patterns is proposed in *TIMES* (Chai 1998). This system makes use of a graphical user interface to create an initial set of rules for the target information contained in the training text. A rule is represented as a "transition" made of two nodes (usually, noun phrases) and a relation between two nodes (usually, verb phrases or prepositions). The sense information for each headword in a noun phrase is provided by WordNet. WordNet is also used in the process of generalizing the initial set of rules. The rules learned by *TIMES* are single slot. Figure 2.6 shows a sample set of extraction rules which have been created based on the same relevant sentence but with different type of target information. The left hand side of each rule specifies conditions (a conjunction of entities each one consisting of three fields: a variable to be instantiated by a new phrase, a concept, and the type of the target information); the right hand side specifies actions (identifies the phrase X_i as the target information T_i).

TIMES has a rule generalization engine to control the degree of generalization according to the user's demands. First, each specific rule is generalized to its most general form (i.e., noun entities in the specific rule are generalized to their top hypernym in the WordNet hierarchy); in this situation, recall is as high as possible while precision may be too low. Then, to adjust the precision depend-

R_1: S(X_1,{group,...},COMPANY) $\wedge$ S(X_2,{need,...},none) $\wedge$
S(X_3,{professional,...},POSITION) $\Longrightarrow$ FS(X_1,COMPANY)

R_2: S(X_1,{group,...},COMPANY) $\wedge$ S(X_2,{need,...},none) $\wedge$
S(X_3,{professional,...},POSITION) $\Longrightarrow$ FS(X_3,POSITION)

R_3: S(X_1,{group,...},COMPANY) $\wedge$ S(X_2,{need,...},none) $\Longrightarrow$ FS(X_1,COMPANY)

Figure 2.6: Examples of extraction rules created by the *TIMES* system.

ing on the user's needs, the system automatically adapts the generalization levels for each noun entity by means of the statistical classifier. This tool calculates the relevancy_rate[19] for each object concept and generates its corresponding GT (Generalization Tree). Based on the GT model and according to a threshold predefined by the user, the system replaces two[20] most general concepts by the sets of optimal concepts (concepts whose relevancy_rate is higher than the predefined threshold).

TIMES is enhanced by adding to the semantic generalization a syntactic generalization step. Syntactic generalization is performed by means of a combination function (used to select the appropriate number on entities in a rule), and a permutation function (used to get the appropriate order of entities in a rule).

RAPIER (Califf 1998) is a system inspired by *Inductive Logic Programming* (ILP) that is able to learn IE patterns expressing relational dependencies in CP1 logic. This system overcomes the computational cost problem of ILP systems by using an incremental approach.

RAPIER learns single slot extraction rules. A rule consists of three fields: the pre-filler and post-filler patterns, that are matching conditions for the text preceding and following, respectively, the filler, and the filler pattern which describes the matching conditions for the actual slot filler. The rules contain lexical constraints (exact match words), semantic constraints and/or part of speech on each item for each field. Figure 2.7 shows an example of an extraction rule learned by RAPIER for the Area slot in the computer-related jobs domain. The pre and post filler specify that information to be extracted must be preceded by the word "leading" and followed either by "firm" or by "company". The filler pattern specifies the matching conditions for the information to be extracted that, in this example, are at most two words with a POS tag "nn" or "nns".

RAPIER uses a not pure bottom-up approach because it combines bottom-up approach with a top-down component. For each slot, it begins with the most specific rule-base for that slot and then compacts the rule-base by generalizing

[19] How many times an object concept is activated by an actual relevant object.

[20] Corresponding to the two noun entities in the rule.

```
AREA extraction rule:

    Pre-filler Pattern:
    1)word: leading

    Filler Pattern:
    1)list: length 2

    Post-filler Pattern:
    1)word:{firm,company}
      syntactic: {nn,nns}
```

Figure 2.7: A sample RAPIER rule to extract the Area slot.

the specific rules. To do this, it takes random pairs of rules from the rule-base and applies a beam-search algorithm to find the best generalization of each pair: first, it only generalizes the filler patterns and creates rules with the least general generalization filler pattern obtained and with empty pre and post filler patterns; then, it specializes the resulting rules by adding constraints into the pre and post filler patterns.

RAPIER needs relatively large amounts of training examples, consisting of documents paired with filled templates, even for relatively simple extraction tasks. It also makes use of a part-of-speech tagger (the Brill's tagger) and a domain-independent lexicon (WordNet).

In order to ease the expert's task of annotating the training corpus, *active learning* methods have been proposed. An active learning system is able to influence the examples it is given, for example, by selecting the subset of examples, from a set of unsupervised examples, that will be the most informative. Such approach is known as *selective sampling* (Cohn et al. 1994; Lewis and Catlett 1994). The basic idea of this approach is starting with a small set of annotated examples and a large set of unannotated examples, the learner attempts to select additional examples for annotation that are likely to be the most useful. Some works that have applied selective sampling to IE pattern acquisition are (Soderland 1999), (Thompson et al. 1999) and (Turmo 2002).

2.3.2 Semi-supervised Approaches

All these systems obtain IE patterns without the hard work of writing them manually. However, they need an annotated training corpus. This too is tedious work

that must be done by a human expert. An important improvement in the process of acquiring IE patterns would be to avoid the need for an annotated corpus.

An early attempt to elude this problem is AUTOSLOG-TS (Riloff and Shoen 1995), which obtains conceptual patterns for IE using a previously classified training corpus without text annotations. This system is an extension of AUTOSLOG, described above, in which the annotated corpus is substituted by a set of unannotated texts containing positive examples. In this approach, an expert has to validate the set of patterns learned and indicate the information they are able to extract.

More recently, systems have also been proposed which avoid the task of fully annotating the corpus (Brin 1998; Collins and Singer 1999; Riloff and Jones 1999; Agichtein and Gravano 2000; Yangarber 2001). All of them are based on the use of a set of seed examples from which they learn some context conditions that then enable them to hypothesize new positive examples, from which they learn new context conditions, and so on. This approach is known as *bootstrapping*.

In IE, Riloff and Jones (1999) present a method for simultaneously learning a semantic lexicon and a set of patterns for one slot of the output template. The system starts with a small set of words (*seeds*) that belong to the semantic category to be extracted. Occurrences of these are then found in the unannotated corpus and extraction patterns are inferred based on the surrounding context. The set of patterns is used to extract new words that belong to the semantic category, and the process is repeated.

DIPRE (Brin 1998) is another system for acquiring patterns which is able to extract binary relations from Web documents. Very simple patterns are learned from a set of seed word pairs that fulfil the target relation (for example, Company - Location). Examples are used to search Web pages for pieces of texts where one word in the relation appears very close to the other. In this case, a pattern is created which expresses the fact that both semantic categories are separated by the same lexical items that separate the example seed words in the piece of text found. The set of patterns obtained from the example relations are used to find more pairs of related words and the process is repeated. The SNOWBALL system (Agichtein and Gravano 2000) is a further development of this idea which uses statistical techniques to avoid the formation of low precision patterns.

Finally, EXDISCO (Yangarber et al. 2000a; Yangarber 2001) is a bootstrapping method in which extraction patterns are learned from an initial set of IE patterns whose application in a text indicates that the text is suitable for extracting a target event. By applying the set of seed patterns, the unannotated corpus is divided into relevant and irrelevant texts. An exploratory search for patterns correlated with the set of relevant texts, allows one to guess new extraction pat-

terns that can be used to search new relevant documents, and so on. Unlike the previous approaches, a human expert has to indicate which slots of the output template are filled by each learned pattern.

In spite of the fact that the bootstrapping approach is very appealing due to its reduction in handcrafting, it does present some problems. The main disadvantage of bootstrapping approaches is that, although the initial set of seed examples could be very reliable for the task in hand, the accuracy of the learned patterns quickly decreases if any wrong patterns are accepted in a single round. Systems based on bootstrapping techniques must incorporate statistical or confidence measures for patterns in order to limit this problem. Another drawback is that these systems need a large corpus (of the order of thousands of texts), which is not feasible in some domains. Finally, the bootstrapping approach is also dependent on the set of seed examples that are provided by the expert. A bad set of seed examples could lead to a poor set of extraction patterns.

Bootstrapping is closely related to *Co-training* (Blum and Mitchell 1998). Though the approaches are not identical Co-training theory could be useful for providing a theoretical basis in a Probably Approximately Correct (PAC) learning framework for the approaches described above. Blum and Mitchell (1998) present formal results for learnability from a small set of examples and a large set of unlabelled examples where the examples can be described using two different views (sets of features describing the examples) that can be used independently for learning the target concept[21]. Two learning algorithms are started independently, each one on a different view of the examples. When the set of labelled examples has been exhausted, the concept learned from one view is used to label unlabelled examples that will be used for learning in the other view. In the systems presented above for IE tasks, bootstrapping is always between two sides: words in a semantic category and patterns for extracting words in this category (Riloff and Jones 1999); examples of a binary relation and extraction patterns for this relation (Brin 1998; Agichtein and Gravano 2000); relevant documents and patterns to extract information from relevant documents (Yangarber et al. 2000a; Yangarber 2001). Learning (or expansion) takes place on both sides. Each side can be conceptualized as a view in the Co-training framework.

While Co-training has been successfully applied in NLP, especially in IR, the only work we know to date of an explicit application of this framework to an IE task (Named Entity task, not Scenario Template) is the work of Collins and Singer (1999). They apply the Co-training approach that is improved using *boosting* techniques (Schapire 1990). Although their approach is well grounded formally, using it for the Named Entity task seems too ad-hoc, especially because

[21] Plus other theoretical conditions, for example, independence of both sets of features.

of the preprocessing of data and the selection of the set of features to represent the examples.

In addition to the issue of handcrafting the set of IE patterns or annotating a corpus for learning, there is another issue that makes the process of building IE systems costly, namely the construction of a semantic lexicon. Semantic knowledge helps define more specific constraints on the information to be extracted by an IE pattern, especially in IE on non-structured texts where syntax is usually not enough to proportion a reliable extraction of information. Most systems that use semantic knowledge rely on a semantic ontology that is created by a human expert. The only exceptions we know of are Riloff and Jones (1999), reviewed above, which learns semantic categories along with IE patterns, and Bagga *et al.* (1997), which relies on the use of WordNet (Miller 1995) for aiding the expert in building IE patterns.

This thesis presents a new method that facilitates the process of building an IE system by limiting the expert's effort to defining the task and supervising the final results. This involves the acquisition of extraction patterns without annotated corpora and with a general domain ontology - WordNet in our case. This is achieved through the ESSENCE method which applies ELA, a new learning algorithm for acquiring IE patterns.

Chapter 3

The ESSENCE method

The ESSENCE method is intended to reduce human expert effort when building IE patterns. This goal is achieved by means of a pattern generalization (learning) algorithm, called ELA, which delays the expert intervention as far as possible and simplifies the amount of information he/she has to deal with. Nevertheless, an expert is still required in order to validate the results and specify the type of information to be extracted.

The method we propose makes use of NLP tools such as a semantic ontology and a syntactic parser. In order to make the ESSENCE method as portable as possible, the knowledge sources and NLP tools it uses must also be as portable as possible. For this reason, WordNet[1], a general-purpose lexicon, has been selected. WordNet offers lexical, syntactic and semantic information which is useful in the generalization process. Although the suitability of WordNet to IE tasks is controversial, we found it effective in the experiments we conducted. In section 7.1 we point out some reasons that could explain the adequacy of WordNet. However, the lack of covering for some domain specific words such as entity names can be overcame by using domain linguistic modules such as gazetteers and domain specific word lists. Portability of the syntactic parser is assured because it is domain independent.

The distinctive features of ESSENCE can be synthesized as:

- The training corpus has no annotations, neither syntactic tags nor semantic tags, but must include typical examples of text carrying information that is to be extracted.
- Human intervention is restricted to the *task definition* and *typification* and final *validation* of patterns.

[1] http://www.cogsci.princeton.edu/∼wn

```
Crash_Event:
  Crash_Site:     Dominican Republic
  Crash_Date:     06-02-1996
  Aircraft:       757
  Airline:        Alas Nacionales
  Manufacturer:   Boeing
  Departure:      Dominican Republic
  Destination:    -
```

Figure 3.1: Example of filled event template.

- For the generalization (learning) process a semantic hierarchy is needed. ESSENCE makes use of WordNet, which is able to cover multi-domain vocabulary, instead of a hand built semantic hierarchy tailored to each new domain.

Each component step of the ESSENCE method, depicted later in figure 3.3, is described in the following sections.

3.1 Task Definition

Since IE is oriented to a specific task, the expert has to define the type of information to be extracted. The *Task Definition* module has been designed to assist the expert in this work. Basically, for each slot of the output template (see sample template in figure 3.1) he/she defines a set of *Extracting Synsets* and a set of *Context Keywords*. These sets are necessary for the ESSENCE method, and thus the task definition step (depicted in figure 3.2) must be done in advance.

Extracting Synsets

The set of *Extracting Synsets* represents the semantic values that an output slot can take. A synset is a number that identifies a concept in the WordNet ontology. For example, consider the `CRASH-SITE` slot in the output template for the aircraft crash domain. In this case, following the requirements of the task, we could assign to this slot the values:[2] 6666185, 6359477, 6668569, 2661119, 6691504, 6667942, corresponding respectively to the concepts "body_of_water", "region", "ground", "facility", "geological_formation" and "terra_firma". The

[2]The synset numbers are extracted from version 1.6 of WordNet.

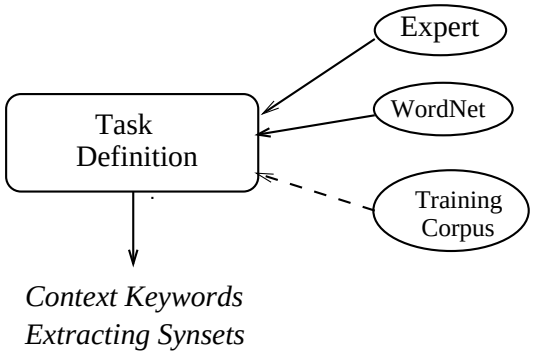


Figure 3.2: The Task Definition step.

meaning of this assignment is that only words with these semantic values (or hyponyms of these values) in WordNet can be used to fill the slot.

In case the expert is not familiar with WordNet, he/she is aided by an easy-to-use interface. This tool asks for examples of words that can fill a slot. From these words, the tool shows a set of synsets that correspond to the union of synsets defined in WordNet for the words. The expert can remove synsets that do not correspond to valid senses for filling the slot. For example, the expert may provide the word “plane” meaning an airplane. In this case, the tool will provide all synsets (along with the description) corresponding to “plane”. Now the expert can remove senses (synset numbers) that are not appropriate for filling the target slot.

Then, for each of these synset values, the tool shows its hypernym synset (along with its description or textual gloss) in WordNet and asks the expert if this synset is a better description for the slot. In our example, the tool will show the hypernym of synset 2014460 for “plane”. The new synset, number 2170808, represents the concept “aircraft”. This is a more general sense and allows the slot to be filled, for example, with helicopter crashes or other items that were not covered by the synset for plane. If the expert accepts the new synset for filling the slot, the process is repeated showing a new hypernym and so on until the expert rejects the proposal.

In some cases, the information to be extracted is expressed in linguistic terms not covered by WordNet. For example, in the aircraft crash domain, the specific aircraft model or the airline involved in the crash. In these cases, domain specific Named Entity recognizers are responsible for the identification of the semantic category for the entity. Each of these modules tags entities not covered by WordNet with a specific semantic tag. When it is possible, the specific semantic tag

must be a compatible WordNet synset. For example, the entity names provided by a gazetteer can be tagged with the synset number in WordNet corresponding to the concept "region", which is defined as an Extracting Synset. When this is not possible, the expert must define a new specific semantic tag for the category identified by the domain specific recognizer, and use this tag as the Extracting Synset for the slot.

Context Keywords

The *Context Keywords* for a slot is a list of words that commonly appear together with the type of information to be extracted. They therefore play the role of trigger words for selecting sentences that might contain relevant information. Context keywords must be nouns or verbs, and their POS must be explicitly given[3]. In order to obtain the keywords, the expert can guess some words or browse the corpus to find an initial set of words. This set of words is automatically extended by finding in WordNet all synonyms and hyponyms of each word in the initial set. The resulting set is filtered by the expert and it is finally completed by means of a morphological tool which generates all the forms of each word depending on its syntactic category.

The reason for using words instead of synset numbers, as in the case of the extracting synsets, is a practical one: to avoid the semantic tagging of the whole corpus, the Relevant Sentence Filtering process (relevant in the sense that the sentence carries a context keyword), described later in section 3.4.1, limits semantic tagging to relevant sentences. Note that the filtering process is prior to semantic tagging, and therefore we have to select relevant sentences on the basis of the presence of words rather than the presence of meanings.

In chapter 5, an example of the whole process of task definition that includes the generation of the sets of context keywords and the sets of extracting synsets is shown.

3.2 Preparing the Training Corpus

The ESSENCE method is designed to extract target information from free texts. It starts with an untagged corpus of a particular domain, including prototypical texts for that domain. The set of texts will be used for training the system and must contain relevant texts to the IE task. This means that the training corpus must contain instances of events to be extracted although instances of irrelevant events can also be present. In ESSENCE, even if the expert supplies the system

[3]Sometimes a word is a useful trigger word as a verb but not as a noun.

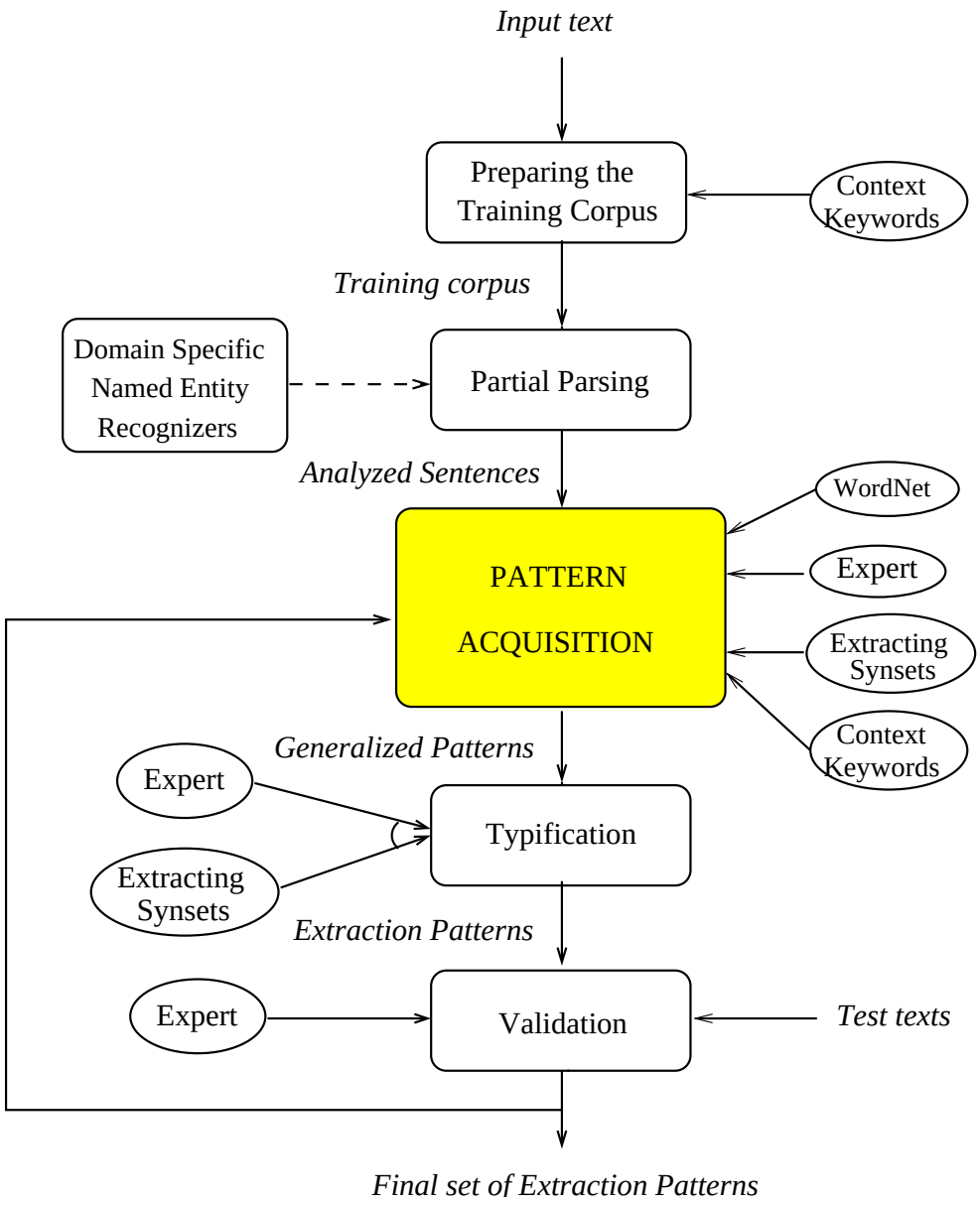


Figure 3.3: Overview of the ESSENCE method.

with a corpus containing non-relevant documents, they will be automatically excluded because usually they will not contain sentences with context keywords defined by the expert. This is an implicit text-filtering process.

ESSENCE has proved experimentally to be useful for acquiring IE patterns when working with only 100 texts for training (about 750 words per text) in an IE task in the aircraft crashes domain (see results in section 5.3). This feature distinguishes our approach from the bootstrapping approaches described in section 2.3.2.

3.3 Partial Parsing

Once the training corpus is available, major syntactic constituents for each sentence in the text, such as noun groups, verb groups or prepositional groups, are identified by the *Partial Parsing* module. In ESSENCE this task is performed by an extended version of MARMOT, a shallow syntactic parser developed by the NLP laboratory at the University of Massachusetts[4].

In order to include MARMOT in the ESSENCE method, we had to extend it in the following ways:

- **Broaden the Syntactic Dictionary:**

  MARMOT was designed to use a domain specific dictionary. As we are interested in making ESSENCE as portable as possible, we extended the base dictionary of MARMOT with syntactic knowledge drawn from WordNet. Although WordNet was not designed primarily for describing syntactic information, we selected it in order to maintain consistency because the same tool will be used later in the generation of IE patterns. In order to resolve tagging ambiguities, some heuristics were added to MARMOT to determine the right tag.

- **Extend the general-purpose partial lexicons:**

  MARMOT also has two partial lexicons: one contains known abbreviations and the other is a phrasal lexicon. Both lexicons help the process that resolves sentence boundaries. The use of these lexicons might not be limited to one domain if they contained general-purpose expressions, for example, “Mr.” or “Cmdr.”, the abbreviations of “mister” and “commander” respectively, and considering “take off” as a single unit, are all of general use.

[4] http://www-nlp.cs.umass.edu/

The initial list of abbreviations and the phrasal lexicon proved to be too simple. For this reason, they were extended using a table of standardized abbreviations and a thesaurus, respectively.

In addition to these domain independent modifications, sometimes MARMOT has to be tuned for the specific domain of the IE task. Tuning mainly consists of adding some domain specific vocabulary. Although WordNet offers a large covering across domains, there are several domain specific words that do not appear. For example, in the scenario concerning aircraft crashes, names of artifacts, proper nouns, organization names and some locations were not found. It is worth mentioning that most of these domain specific names were multi-word expressions, and it is very important to recognize each multi-word expression as a single unit for further processing. Specific Named Entity recognizers, gazetteers or specific word lists resolve the lack of covering of domain specific vocabulary. All the entities recognized by using a given Named Entity module will be assigned to the same semantic category and will be tagged with a special semantic tag. Semantic tagging is completed later (see section 3.4.3).

At the end of this module we have a partially parsed corpus that feeds the *Pattern Acquisition module*.

3.4 Pattern Acquisition

Multi-module *Pattern Acquisition* represents the core of the method: it produces generalized patterns from a set of analyzed sentences. *Pattern Acquisition* comprises six sub-modules, as depicted in figure 3.4.

3.4.1 Relevant Sentence Filtering

Relevant Sentence Filtering allows the system to focus only on sentences that contain information related to the task in hand; those sentences that contain one or more context keywords with the right part of speech are considered relevant. From the whole set of sentences the system only keeps the relevant ones.

Relevant sentence filtering is performed at this point, after partial parsing, because it requires the part of speech attached to each word in the sentence to be known.

3.4.2 Windowing

From each *relevant sentence* we build a parameter-sized context window. A window is the context surrounding an occurrence of a context keyword, and the size

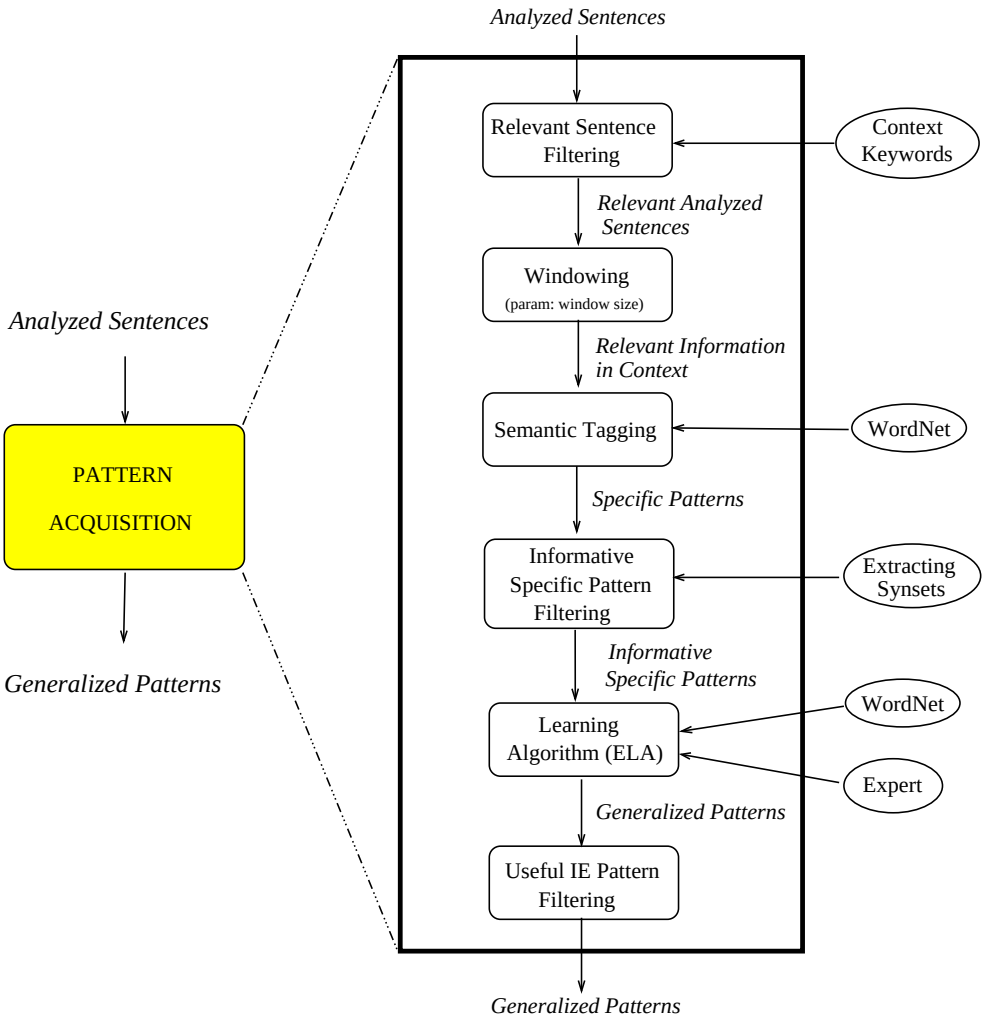


Figure 3.4: Detailed description of the *Pattern Acquisition* module.

```
Specific pattern: ( number_of_sentence
                    list_of_syntactic_groups )

Syntactic group : ( syntactic_category_of_the_group
                    list_of_constituents_of_the_group )

Constituent :     ( syntactic_category
                    word
                    set_of_synset_numbers )
```

Figure 3.5: Template for specific patterns.

(width) of a window is the maximum number of syntactic groups it includes[5]. The keyword will be in the center of the window and the context will be formed by $\lfloor \frac{width-1}{2} \rfloor$ syntactic components before the keyword and $\lceil \frac{width-1}{2} \rceil$ syntactic components after the keyword.

Some sentences containing context keywords are too large and include a large amount of information. Part of this information is usually irrelevant to the task and slows down the generalization process. The windowing procedure attempts to rid of from long sentences information that is not related to the keyword, in the hope that the context windows will retain the target information.

3.4.3 Semantic Tagging

This step tags the headword of each syntactic group with WordNet's synsets. Synsets are collected by searching in WordNet for all possible meanings of the headword corresponding to the POS assigned to the word. Note that no disambiguation procedure is performed at this point.

The *Semantic Tagging* module implements the idea that the meaning of the headword of a syntactic group represents the meaning of the whole group. The head of a group is a verb or a noun. This is especially convenient for our approach because it allows us to generalize semantic senses using WordNet, and establishes the hyponymy/hypernymy hierarchical relations needed for the generalization process.

Semantic tagging using WordNet is not performed on groups where the head has been recognized by a Named Entity module in the *Partial Parsing* step (see section 3.3) as domain specific vocabulary. When a headword has been recognized neither by WordNet nor by domain specific NE modules, the semantic tag

[5]It is a maximum number because a window does not exceed sentence boundaries.

Sentences:

1- The flier whose Navy F-14A fighter plunged into a Nashville suburb *on Monday, killing himself and four other people,* **crashed** *another jet into the sea last April.*

53- Commerce Secretary Ron Brown and 32 others on a Balkan trade mission were presumed killed *when their plane* **slammed** *into a Croatian hillside during heavy storms Wednesday.*

```
(1 ((PP ((PREP ON NIL) (DATE MONDAY DATE)))
    (PP ((PREP KILLING NIL) (NPST HIMSELF NIL)
         (CONJ AND NIL) (NOUN FOUR (9896114))
         (ADV OTHER NIL)
         (NOUN PEOPLE (5957883 6069040 6080290 5976176))))
    (VP ((VERB CRASHED (1342612 1431218 1076088 1379139
          1076294 1813216 1378886 1076442 1035304 303160
          12099))))
    (NP ((NPST ANOTHER NIL)
         (NOUN JET (2875044 5531909 10710122 2717915))))
    (PP ((PREP INTO NIL) (NPST THE NIL)
         (NOUN SEA (6781925 9922052 7845203))))
    (PP ((DATE LAST_APRIL DATE)))))

(53 ((PP ((CONN WHEN NIL)))
     (NP ((NPST THEIR NIL) (NOUN PLANE (2174460 9985988
          10046013 3137218 3136725))))
     (VP ((VERB SLAMMED (847148 846778 1295221 847023))))
     (PP ((PREP INTO NIL) (NPST A NIL)
          (NOUN CROATIAN (7052505))
          (NOUN HILLSIDE (6724431))))
     (PP ((PREP DURING NIL) (NOUN HEAVY (7314549 4553181))
          (NOUN STORMS (7803078 10069810 627161))))
     (PP ((DATE WEDNESDAY DATE)))))
```

Figure 3.6: Examples of specific patterns derived from sentences in the aircraft crash domain. Context window width is set to 6.

assigned to it is NIL.

The output of this module is a set of context windows with their syntactic constituents semantically tagged. This set is called the set of *specific patterns* which will be used as observations for the learning algorithm described in the next chapter. The structure of specific patterns is depicted in figure 3.5.

Figure 3.6 shows two sentences extracted from the corpus used in our experiments. These sentences have been selected because they contain context keywords, in this case "crashed" and "slammed". The figure also shows the corresponding specific patterns obtained from these sentences after parsing and windowing have been performed. The value for the $width$ parameter is 6. The headword of each syntactic group recognized by WordNet is tagged with all senses of the word. Groups recognized by auxiliary NE modules are semantically tagged with specific semantic tags (in the example, DATE).

3.4.4 Informative Specific Pattern Filtering

Once the semantic tagging has been performed, a second filter is applied to the set of specific patterns. This filter will retain only specific patterns that contain at least one group with semantic values that can be used to fill one slot of the output template. The semantic values that can fill a slot are the *extracting synsets* for the slot or their hyponyms. The reason for doing this filtering is that specific patterns that do not carry information to be extracted are useless for learning relevant IE patterns for the task defined.

We call those groups in a generalized or specific pattern that are semantically tagged with an extracting synset (or a hyponym of one) for one slot of the output template *informative groups*.

3.4.5 Learning Algorithm

From the set of specific patterns the *Learning Algorithm* generates a set of *generalized patterns*. A generalized pattern is a set of syntactic and semantic constraints that are fulfilled by a set of specific patterns. These constraints require the presence of precise syntactic groups (verb groups, noun groups or prepositional groups with the right preposition) with precise semantic tags.

Syntactic patterns are able to recognize exactly the same information they hold. Therefore, they are patterns with very reduced applicability. Generalization is required to make specific patterns applicable to new texts in the domain.

The *Learning Algorithm* module is detailed in chapter 4.

3.4.6 Useful IE Pattern Filtering

This last filtering process allows the system to discard any useless patterns that the generalization procedure may generate. Here, generalized patterns with no generalized semantic tags are removed. These generalized patterns can appear in some cases because they seemingly cover several specific patterns that in fact correspond to literally repeated pieces of text.

In addition, the expert can define a minimum number of cases that each pattern must cover. In case that one pattern does not achieve this covering, it is also discarded.

3.5 Typification

A generalized pattern requires the presence of precise syntactic groups with precise semantic tags in sentences. Some of the required groups are *informative* (see section 3.4.4). The remaining groups are *contextual*. The presence of the latter increases the precision of the pattern, but no information is extracted from them.

In order to extract the target information, a link must still be provided between the information that informative groups extract and the output slots.

The *Typification* module "gives names" to the different pattern groups, indicating the type of information they will extract. It then determines which extracted information will fill which slot of the output template. A typified generalized pattern is in effect an IE pattern.

The typification process is usually automatic. In the task definition step, the expert defines the set of *extracting synsets* for each slot of the output template, that is, the semantic values each slot can take. When the semantic value of a group in a generalized pattern is the same as (or a hyponym of) the semantic values that one slot can take, the process links that output slot to that group, indicating that the group will extract such information. The only exception to this is when the semantic value of a group is an extracting synset (or a hyponym of) for *more than one* output slot. In such cases, the expert must decide which slot the group will fill.

Figure 3.7 describes the structure of typified generalized patterns, that is, final IE patterns. Figure 3.8 shows an example of such patterns. The example pattern requires: (1) a verb group with its head in set 1 of context keywords (defined by the expert in our experiments to extract crash locations and crash dates), (2) a preposition group with the preposition `"INTO"` and a WordNet "geological formation" head, and (3) a preposition group containing a date. The pattern has been typified to extract information for the `CRASH-SITE` and the

```
IE pattern   : ( number_of_groups_in_the_pattern
                 cost_of_the_last_relaxation
                 list_of_constraints )

Constraint   : ( syntactic_category_of_the_constraint
                 list_of_constituents_of_the_constraint
                 slot_to_be_filled_by_the_constraint )

Constituent  : ( syntactic_category
                 words_in_the_training_set_matched
                 synset_number_or_specific_semantic_tag )
```

Figure 3.7: Template for typified generalized patterns.

```
((3 6)
   ((VP ((VERB (CRASHED SLAMMED) KEYWORD-1))
           CONTEXTUAL-GROUP)
    (PP ((PREP INTO NIL) (NOUN (SEA HILLSIDE) (9457)))
           CRASH-SITE)
    (PP ((PREP NIL NIL) (DATE (LAST_APRIL WEDNESDAY) DATE))
           CRASH-DATE))
)
```

Figure 3.8: Example of a typified generalized pattern, i.e., a final IE pattern. The last tag of each group indicates which slot of the output template will be filled with information extracted by that group.

CRASH-DATE slots. The remaining group, that which requires the context key, is typified as a *contextual group*.

3.6 Validation

Finally, the set of extraction patterns obtained must be validated. The validation process consists in applying the IE patterns to extract the information from a test corpus.

For some domains, it is possible to compare the information extracted by the system with a database containing the correct answers, to evaluate the system performance. Examples of these domains are those used in MUC competitions.

Results on testing are then analyzed by the expert. If results at this stage were

not satisfactory, they could still be useful as feedback for a new execution of the learning process. For example, the expert could resolve to change the window width parameter or enlarge context keyword lists.

Chapter 4

The ESSENCE Learning Algorithm: ELA

The learning algorithm is the core of the ESSENCE method. Inspired by ML techniques, ELA explores the set of specific patterns provided by the *Semantic Tagging* module (see section 3.4.3) in order to find regularities that will be used to build a set of generalized patterns. The proposed learning algorithm has been designed specifically for IE tasks. It builds extraction patterns from a set of observations (unannotated instances) rather than from a set of labelled examples, thereby becoming closer to a text data mining algorithm than to a classical supervised learning algorithm.

In ML approaches for learning IE patterns, the expert usually has to provide the system with an annotated corpus containing examples of the type of information to be extracted. Such examples not only consist of a set of particular sentences that contain information to be extracted, but also an indication of the specific information to be extracted from the sentence and the slot it must fill in the output template. The main drawback of these approaches is the cost associated with annotating a corpus in such a way.

One way to overcome this problem is to use *unsupervised learning* algorithms which do not learn from examples but from observations. In IE, an observation is defined as an example without annotations referring to information to be extracted.

Unsupervised learning has mainly been used for *concept formation* based on similarity between observations. Specifically, Michalski and Stepp (1983) proposed the term *conceptual clustering* as the task of discovering 'understandable' patterns in data. This topic is closely related to research done on *clustering* algorithms in statistics (Kaufmann and Rousseeuw 1990), which group objects or observations that are close (given a similarity measure) in the feature space that

describes them. Currently, these techniques are widely used in *Data Mining* and *Knowledge Discovery*.

The goal of these approaches is to build sets of observations (called *clusters* or *concepts*) by analyzing similarities between observations. Similar observations fall into the same cluster. Most algorithms describe clusters or concepts learned by means of a *prototype* (the average observation inside the class) or by defining a set of properties that every member of the cluster fulfils (this is also known as a *covering description*).

In order to apply an unsupervised learning algorithm to the acquisition of IE patterns, we assume that in Natural Language, the structure of the sentence and the semantics of words allow the grouping of pieces of text by the information to be extracted. This is an assumable hypothesis that is implicit in the use of patterns for extracting information. In fact, an IE pattern can be considered the definition of a cluster (a covering description).

ELA builds clusters starting from a seed specific pattern (our observations), and iteratively adds to the cluster the closest observation satisfying a set of conditions until no new observations can be added. The measure that determines the closest observation to a cluster is defined in the next section. The main advantage of the approach we present is that only a corpus representative of the domain is needed. This corpus contains no annotated sentences.

4.1 Relaxation

In our approach, an IE pattern is composed of a set of constraints represented as *noun groups*, *verb groups* and *preposition groups*. Each group is tagged with a WordNet synset that is determined by the generalization procedure described below. These groups specify which type of groups a sentence must contain in order to satisfy the IE pattern.

Information from a sentence is extracted by an IE pattern when the sentence contains, for each group in the pattern, a group that matches with it. We say that a group in a sentence *matches* a group in the IE pattern when the head of the sentence's group has a semantic tag that is a hyponym in WordNet of the synset of the pattern's group[1]. In this way, information to be extracted is contained in those groups in the sentence that match informative groups in the IE pattern.

As described in section 3.4.3, a specific pattern is a windowed sentence com-

[1]Additional constraints can be added in order to consider when two groups match. In our experiments we also require the same preposition in preposition groups. In further experiments we could also require the same voice for verb groups, the same number for noun groups, or the same relative position in the pattern.

posed of syntactic groups, where the headword of each syntactic group is tagged semantically with the corresponding set of WordNet synsets. This structure is too specific as an IE pattern because it describes a set of constraints that can only be satisfied by the sentence that originated it. For effective IE systems, we need to find more widely applicable patterns. A *generalized pattern* is a candidate for being an IE pattern describing a set of constraints that are fulfilled by several specific patterns (that is, for each group of the generalized pattern, there exists a group in each specific pattern that matches it). If this holds then we say that all these specific patterns are *covered* by the generalized pattern.

The algorithm we propose (ELA) follows a bottom-up approach. The way to obtain a generalized pattern consists of initially setting it to a randomly selected specific pattern and then repeatedly generalizing it in order to cover a new specific pattern with each iteration. Generalization is achieved *by relaxing the semantic tags* associated with the groups of the pattern and/or *by removing groups* from the pattern when the former is not possible.

Relaxation of a semantic tag in the generalized pattern is performed in order to find a semantic description of the group that could also cover a group in the new specific pattern (allowing both groups to match). For example, assume that the generalized pattern has a group with the semantic tag representing an AVALANCHE and the specific pattern has a group tagged semantically as a CRASH. If the semantic ontology has defined both concepts as hyponyms of ACCIDENT, then both groups can be covered if the semantic tag of the generalized pattern is relaxed (generalized) to ACCIDENT.

The selection of the new pattern to be covered each time is done by searching the whole set of specific patterns for the one *closest* to the generalized pattern in hand. In ELA, the closest pattern is the specific pattern that requires a *minimum generalization* of the generalized pattern in order to cover also the specific pattern (in other words, the specific pattern that forces a minimum generalization of the generalized pattern in order to be covered by it). As we have seen, generalization is done by relaxing semantic tags associated with the groups in the pattern and/or by removing groups from the pattern when the former is not possible. The measure of the cost of generalization, our *relaxation measure*, takes into account the number of groups that remain (a measure of the removed groups), and the generalization that has been done in the semantic ontology in the groups that remain. Formally, the relaxation measure is defined as follows:

$$Relax(x, y, M) \quad = \quad w_1 \ \ (LW - |M|) \quad + \quad w_2 \sum_{<i,j>\in M} MC(x_i, y_j) \quad (4.1)$$

where M is a coherent[2] set of pairs $< i, j >$ indicating that group number i of the *generalized*[3] pattern x (represented by x_i) matches group j of the *specific* pattern y (y_j), LW is the maximum number of groups in a pattern (the width parameter in section 3.4.2), $MC(x_i, y_j)$ is the cost of generalizing groups x_i and y_j for allowing their match, and finally, w_1 and w_2 are parameters given by the user in order to balance the relative influence of the cost of generalizing semantic tags with respect to removing groups in the $Relax$ function.

This formula measures the cost of generalizing pattern x in order to cover pattern y for the set of matching groups defined in M. The fact that removing a group is similar to generalizing that group to the top node of the semantic hierarchy could give us some clues for assigning values to w_1 and w_2. Assuming that the average depth of a synset in WordNet is 7, setting w_1 to 7 and w_2 to 1 will give approximately the same influence to both terms in the equation. Giving values higher than 10 to w_1 when w_2 is set to 1 will facilitate the maintaining of groups in the generalized pattern. Values for w_1 less than 5 when w_2 is set to 1 will facilitate the removal of groups in the generalized pattern. Note that, $Relax$ being a comparative measure and w_1, w_2 relative values (for example, values $w_1 = 2$ and $w_2 = 1$ would lead to the same ranking of similarity among patterns as setting $w_1 = 2k$ and $w_2 = 1k$ for any k), the measure could be formulated using only one parameter, for example α ($0 \leq \alpha \leq 1$), and replacing w_1 by α and w_2 by $(1-\alpha)$. Nevertheless, we keep the original formulation because it is easier for the expert to work with relative values than to work with the α parameter.

The generalization cost for matching groups x_i from the generalized pattern and y_j from the specific pattern, $MC(x_i, y_j)$, is measured by counting the minimum number of levels to climb in the ontology from the semantic tag of x_i, to find a synset that covers both groups. In cases where headwords of x_i and y_j were *context keywords* for the same output slot, or were recognized by the same domain specific named entity module, the generalization cost is 0. Formally,

[2]That is, i and j cannot appear more than once in the set and both values are less than or equal to the number of groups in patterns x and y respectively. In addition, the syntactic category (and the preposition in preposition groups) for x_i and y_j has to be the same.

[3]We consider a seed pattern also as a generalized pattern.

$$MC(x_i, y_j) = \begin{cases} 0 & \text{if } \exists\, k \;\; (h(x_i), h(y_j) \subseteq CK_k \vee \\ & \qquad S(x_i), S(y_j) = NE_k \,) \\ \min\limits_{\substack{k \in S(x_i) \\ l \in S(y_j)}} \; D(k) - D(mchper(k,l)) & \text{if } \exists\, mchper(k,l) \;\wedge \\ & \qquad D(mchper(k,l)) > \theta_1 \\ \infty & \text{otherwise.} \end{cases} \tag{4.2}$$

where $D(k)$ is the depth in the WordNet ontology of sense k, $h(x_i)$ is the headword of group x_i, CK_k is the set of context keywords for the slot k of the output template, $S(x_i)$ is the set of semantic tags of the headword in group x_i, NE_k is the semantic category attached by the named entity module k, $mchper(k,l)$ is the minimum common hypernym in WordNet for synsets k and l, and finally, θ_1 is a threshold parameter for preventing generalized patterns with the excessively general concepts existing in the first levels of the WordNet ontology (where all concepts do match).

Now, finding the closest pattern for generalizing pattern x is defined as finding the pattern y with a set of matching groups M that minimizes the $Relax$ measure. That is, if we define $BestM(x,y)$ as the set M of matching groups that minimizes the $Relax$ measure between x and y patterns (in cases where more than one set M minimizes this measure, we take one of them randomly),

$$BestM(x,y) = \arg\min_{M} \; Relax(x,y,M) \tag{4.3}$$

the closest pattern to pattern x (which we call $CP(x)$) is defined as the following:

$$CP(x) = \arg\min_{y} \; Relax(x,y,BestM(x,y)) \tag{4.4}$$

In cases where more than one pattern minimizes this measure, one of them is selected randomly as the closest pattern to x.

Minimum generalization of generalized pattern x to cover specific pattern y is achieved by performing a minimum generalization of all groups in the generalized pattern that match the specific pattern (following $BestM(x,y)$), and by removing the other groups. *Minimum generalization of a group* is performed by setting the semantic tag of the generalized group to the $mchper(k,l)$ that minimized $MC(x_i, y_j)$. Using these definitions, pattern x is transformed into a more general pattern by performing a minimum generalization of the generalized pattern to cover the closest specific pattern $CP(x)$.

In order to avoid an excessive generalization when attempting to cover the closest pattern, we define the θ_2 parameter. When the $Relax$ value for the closest pattern is higher than a threshold parameter θ_2[4], no further generalization of the generalized pattern is performed.

It is worth noting that minimum generalization involves an implicit word sense disambiguation (WSD) process. From the set of synsets describing the headword of y_j, we assume that the right one is the closest in the WordNet hierarchy to the headword of the matching group in the generalized pattern (note that the matching group in the generalized pattern has only one semantic tag because it results from a previous generalization where disambiguation was implicitly performed). This side effect is not undesirable at all. All specific patterns are extracted from the same corpus provided for one specific domain, contain context keywords and carry at least one group with an extracting synset. We can conclude that to some extent they share the same context. Some successful WSD procedures attempt to find out the sense of one word by assuming that words in the same context conditions will have similar meanings. In some works, for example (Agirre and Rigau 1996), similarity is even measured using WordNet relations.

Figure 4.2 shows a set of generalized patterns obtained from the two specific patterns shown in figure 4.1. Note that all patterns have a relaxation value below 16 because the maximum relaxation allowed, θ_2, has been set to this value. Note also that all of them contain three matching groups because, in the example, we limited the number of matches $|M|$ to this value. Finally, note that each generalized pattern contains a group tagged with the label `KEYWORD-1` rather than a synset number. This represents the constraint that any sentence fulfilling the IE pattern must contain at least one context keyword.

4.2 ELA

The goal of ELA is to find a set of generalized patterns able to extract information relevant to the task. In the training corpus, this information is contained in those groups of specific patterns that are semantically tagged with *extracting synsets* for one slot.

ELA learns a set of patterns that covers information to be extracted. Each pattern is learned in the run of a function called *learn-one-pattern* that takes the set of specific patterns as input and returns a generalized pattern covering part of the information to be extracted. Information covered by this new pattern is marked as extractable and will no longer be considered for learning. The process

[4]That is, when $Relax(x, CP(x), BestM(x, CP(x))) > \theta_2$.

Sentences:

1- The flier whose Navy F-14A fighter plunged into a Nashville suburb *on Monday, killing himself and four other people,* **crashed** *another jet into the sea last April.*

53- Commerce Secretary Ron Brown and 32 others on a Balkan trade mission were presumed killed *when their plane* **slammed** *into a Croatian hillside during heavy storms Wednesday.*

```
(1 ((PP ((PREP ON NIL) (DATE MONDAY DATE)))
    (PP ((PREP KILLING NIL) (NPST HIMSELF NIL)
         (CONJ AND NIL) (NOUN FOUR (9896114))
         (ADV OTHER NIL)
         (NOUN PEOPLE (5957883 6069040 6080290 5976176))))
    (VP ((VERB CRASHED (1342612 1431218 1076088 1379139
          1076294 1813216 1378886 1076442 1035304 303160
          12099))))
    (NP ((NPST ANOTHER NIL)
         (NOUN JET (2875044 5531909 10710122 2717915))))
    (PP ((PREP INTO NIL) (NPST THE NIL)
         (NOUN SEA (6781925 9922052 7845203))))
    (PP ((DATE LAST_APRIL DATE)))))

(53 ((PP ((CONN WHEN NIL)))
     (NP ((NPST THEIR NIL) (NOUN PLANE (2174460 9985988
          10046013 3137218 3136725))))
     (VP ((VERB SLAMMED (847148 846778 1295221 847023))))
     (PP ((PREP INTO NIL) (NPST A NIL)
          (NOUN CROATIAN (7052505))
          (NOUN HILLSIDE (6724431))))
     (PP ((PREP DURING NIL) (NOUN HEAVY (7314549 4553181))
          (NOUN STORMS (7803078 10069810 627161))))
     (PP ((DATE WEDNESDAY DATE)))))
```

Figure 4.1: Initial specific patterns.

Structure:
((number_of_matches measure_of_relaxation)
  list of generalized groups that compose the generalized pattern)

```
((3 9)
   ((VP ((VERB (CRASHED SLAMMED) KEYWORD-1)))
    (PP ((PREP INTO NIL) (NOUN (SEA HILLSIDE) (9457))))
    (PP ((PREP NIL NIL)
         (DATE (LAST_APRIL WEDNESDAY) DATE)))))
((3 10)
   ((VP ((VERB (CRASHED SLAMMED) KEYWORD-1)))
    (PP ((PREP INTO NIL) (NOUN (SEA HILLSIDE) (9457))))
    (NP ((NOUN (JET PLANE) (2174460))))))
((3 4)
   ((VP ((VERB (CRASHED SLAMMED) KEYWORD-1)))
    (PP ((PREP NIL NIL)
         (DATE (LAST_APRIL WEDNESDAY) DATE)))
    (NP ((NOUN (JET PLANE) (2174460))))))
((3 13)
   ((VP ((VERB (CRASHED SLAMMED) KEYWORD-1)))
    (PP ((PREP NIL NIL)
         (DATE (LAST_APRIL WEDNESDAY) DATE)))
    (NP ((NOUN (JET PLANE) (2859872))))))
((3 15)
   ((VP ((VERB (CRASHED SLAMMED) KEYWORD-1)))
    (PP ((PREP NIL NIL)
         (DATE (LAST_APRIL WEDNESDAY) DATE)))
    (NP ((NOUN (JET PLANE) (9457))))))
```

Figure 4.2: Generalized patterns with $|M| = 3$ from specific patterns shown in figure 4.1. Note that the last three patterns match the same groups but with different synset numbers.

is repeated until all information to be extracted from specific patterns has been covered or no more generalized patterns can be learned. This procedure defines a *sequential covering* learning algorithm. Examples of this type of learning algorithms are AQ (Michalski et al. 1986), FOIL (Quinlan 1990) and CN2 (Clark

Algorithm 1 : *learn-one-pattern* function which is called from ELA.

{ **Requires**: Set of specific patterns (SP) from which to learn, and seed specific pattern ($seed\_pattern$) from which to start the generalization process.}

Function learn-one-pattern ($seed\_pattern$, SP) **returns** generalized_pattern
  Set $pattern\_list$ as the list with only $seed\_pattern$
  Set l_pats to the empty list
  while not empty $pattern\_list$ **do**
    Set $current\_pattern$ to the first one of $pattern\_list$
    Remove $current\_pattern$ from $pattern\_list$
    for each specific pattern $sp_i \in SP$ **do**
      Create the minimum generalization of $current\_pattern$ that covers sp_i and that contain at least one informative group of sp_i not extractable by $current\_pattern$. If this generalization exists, store it in $pattern\_list\_aux$
    endfor
    Remove patterns from $pattern\_list\_aux$ with generalization cost higher than θ_2
    if not empty $pattern\_list\_aux$ **then**
      Add $current\_pattern$ to l_pats
      Set $pattern\_list$ to $pattern\_list\_aux$
      Sort patterns in $pattern\_list$ by the relaxation done to obtain them
    endif
  endwhile
  Evaluation of the list of patterns l_pats and selection of the *best pattern*
  if positive evaluation of l_pats
    then return the *best pattern*
    else return NIL
  endif
Endfunction

and Niblett 1989). This approach has been successfully used to generate IE patterns in CRYSTAL (Soderland et al. 1995a) and RAPIER (Califf 1998), among others.

The *learn-one-pattern* function builds each generalized pattern in a bottom-up way starting from a randomly selected specific pattern. At each iteration the pattern is generalized to cover the closest pattern following the $Relax$ measure, until the next closest pattern to be covered involves a generalization cost higher

than θ_2. Since the goal is to cover information contained in groups carrying extracting synsets, a new condition is added to the definition of the closest pattern: the closest pattern is that which minimizes the generalization cost and that contains a group carrying an extracting synset not yet covered by previously learned patterns. This leads pattern generalization to extract relevant information while avoiding redundant IE pattern formation. Algorithm 1 shows the implementation of the *learn-one-pattern* function that generates the minimum generalizations of the current pattern with all specific patterns, and from them attempts to obtain the closest pattern that fulfils the required constraints.

Note that the algorithm stores the history of generalizations from the specific pattern to the last generalization in the *l_pats* list. At the end of the generalization iteration, *l_pats* list must be validated and the best pattern in the list must be selected. This procedure is necessary because ELA does not learn from positive examples but from *possible* positive examples. Note that the set of specific patterns is a set of candidates for positive examples because they contain a context keyword and at least one group carrying an extracting synset for one slot of the output template. Nevertheless, in some cases, the generalization of the current pattern could be done to include a specific pattern that would not be classified as positive if examples were labelled. Sometimes, the seed specific pattern could be irrelevant or even a negative example for extracting target information. Taking this into account, it is necessary to validate and select the best pattern in *l_pats*.

In our method, there are two different ways of doing this. The first way is a visual examination of the list of patterns by the expert. However, this method could be tedious or, even worse, misleading because patterns are not tested on the corpus. A second way is to select from the list the pattern with a highest mixture of *recall* and *precision*, known as *F* (Lehnert and Sundheim 1991). Briefly, while recall measures the covering of IE patterns, precision measures the quality of IE patterns. Both values are expressed as percentages. A 100% recall indicates that all information that had to be extracted was actually extracted. A 100% precision indicates that all information extracted was right. In order to select the best pattern from the *l_pats* list, the most generalized pattern is applied to the training texts. The extracted information is presented to the expert, who marks it as rightly or wrongly extracted.

The expert's report will be used to measure precision for each pattern in *l_pats*. Measuring recall (covering) is harder because it is not possible to know in advance all information that should be extracted from the corpus. Nevertheless, we use a comparative measure of recall, called *Relative Recall*. The most generalized pattern will extract, by definition, more information than the other patterns in *l_pats*. Therefore, the recall of the most generalized pattern will be used to measure recall of the other patterns. The *Relative Recall* of a pattern

Algorithm 2 : ELA algorithm.

{ **Requires**: Set of specific patterns ($specific\_patterns$) containing groups with context keywords and groups with synset numbers equal or hyponym of a extracting synset for the slot for which we try to learn a set of patterns.}

Function ELA ($specific\_patterns$) **returns** set of patterns
  Set $w_1, w_2, \theta_1, \theta_2$ parameters
  Set $generalized\_patterns\_set$ to the empty set
  Set $Seed\_Set$ to $specific\_patterns$
  while not empty $Seed\_Set$ **do**
    $Seed\_Pattern$:=RandomOneFrom($Seed\_Set$)
    gen_pat:= learn_one_pattern($Seed\_Pattern$, $specific\_patterns$)
    Remove $Seed\_Pattern$ from $Seed\_Set$
    if gen_pat **then**
      Add gen_pat to $generalized\_patterns\_set$;
      Mark the groups of specific patterns covered by gen_pat
      Remove from $Seed\_Set$ specific patterns covered by gen_pat
    endif
  endwhile
  return $generalized\_patterns\_set$
Endfunction

is then measured as the percentage of information recovered by it with respect to the information extracted by the most generalized pattern. Finally, the best pattern in the list will be the one with the highest F replacing recall by Relative Recall.

The *learn-one-pattern* function must be called wisely in order to generate a complete set of generalized IE patterns for the current IE task. The point is to call the function with a different seed pattern each time until no specific patterns remain to be used as seed, or until all informative groups of each specific pattern are covered by the current set of generalized patterns. The resulting algorithm is called ELA and is shown in Algorithm 2.

Chapter 5

MUC-like Experiment

In this chapter we present the results obtained testing ESSENCE on a MUC-like task following the MUC guidelines. This first experiment allows us to evaluate the success of ESSENCE in a standard non-structured style corpus. A second kind of experiments, designed to obtain a more solid evaluation of the learning abilities of ESSENCE, will be presented in the next chapter.

Previous to the exposition of the results from the MUC-like experiment, we will describe to some extent the domain and scenario of extraction, how the expert defines the task in the ESSENCE framework, and the values chosen for the parameters of the ESSENCE method.

5.1 The Task and the Data

The set of experiments presented here are intended to test ESSENCE on a MUC domain, concretely, we applied ESSENCE to MUC-7 dry-run texts.

The scenario for the MUC-7 IE task concerned aircraft crash or accident incident reports and updates. The goal was to find out information about aircraft crashes or accidents, such as the location and the date of the accident, the model of the aircraft involved in it, and flight information (such as the departure and arrival information, the owner of the aircraft or the company that manufactured the aircraft).

MUC-7 organization delivered two text sets, one for training and the other for testing purposes. Each set was composed of 100 texts from the New York Times News Service (see figure 5.1 for an excerpt from a news story). Not all the news stories in the corpus describe the crash of an aircraft but all of them mention the word “crash” at least once.

```
<DOC>
<DOCID> nyt960207.0722 </DOCID>
<TEXT>
SEATTLE - It's the phone call no one wants to get, but
everyone knows might come one day.
It came late Tuesday when Boeing got word that a chartered
757 aircraft crashed shortly after takeoff from the
Dominican Republic.  All 189 passengers are feared dead.
The crash, only the second in the history of the Boeing
757, came less than two months after an American Airlines
757 slammed into a mountain as it approached Cali,
Colombia.  Four people survived the Dec.  20 crash that
killed 160 people.  The cause has not yet been determined.
After hearing the news of Alas Nacionales Flight 301
Tuesday night, members of Boeing's Air Safety
Investigation Group monitored the situation throughout the
night and quickly assembled a team of safety experts to be
on standby in case they were needed at the crash scene.
One Boeing air safety investigator was expected to arrive
Thursday in Puerto Plata to assist a team from the
National Transportation Safety Board and the Dominican
Republic in trying to determine why the two-engine jet
crashed.  More Boeing engineers will be called in if
needed.
...
</TEXT>
</DOC>
```

Figure 5.1: An excerpt from a document of the aircraft crash domain.

Specifically, the ESSENCE method was used to extract the following information:

Crash Date: The date of the crash or accident. Examples: "January 17".

Crash Site: The location of the crash or accident. Examples: "Long Island", "Lockerbie".

Aircraft: The aircraft model involved in the accident or crash. Examples: "DC-9", "F-14", "727".

Airline: The owner of the aircraft; it will either be an organization or a person. Examples: "Navy", "Air Force".

Manufacturer: The company that manufactured the aircraft. Examples: “Boeing”, “Cessna”.

Destination: The location to which the aircraft was headed when the accident or crash occurred. Examples: “JFK International Airport”, “Paris”.

Departure: The location from which the aircraft departed. Examples: “Bonn”, “Heathrow Airport”.

Although ESSENCE does not need a corpus with annotations about information that should be extracted (answer keys), MUC competitions deliver them. We use the answer keys *not for learning* but to automatically validate the patterns generated, and thus releasing the expert from this task. Validation is performed with the known measures of recall, precision and their harmonic average, also known as the *F* measurement with the β value set to one (see section 2.1.1 for a description of these measures).

5.2 Applying the Method to the Task

In the first step of the ESSENCE method, the expert must define a set of context keywords for each slot to be filled in the output event template, in this case for crash information slots (which include the `CRASH-SITE` and `CRASH-DATE` slots), and for flight information slots (which include the `DESTINATION`, `DEPARTURE`, `AIRLINE`, `MANUFACTURER`, and `AIRCRAFT` slots).

5.2.1 Crash-information Slots

Context keywords

The set of context keywords for *verb groups* used for the crash information slots (`CRASH-SITE` and `CRASH-DATE` slots) was built up from six concept words selected by an expert and that usually describe a flight crash. These verbs were:

1. CRASH expressing the crash of a flight,
2. FALL describing the fall of an aircraft,
3. DISAPPEAR expressing the disappearance of a flight from radar screens,
4. EXPLODE describing an explosive flight accident,
5. PLUNGE expressing a crash into water, and

6. KILL expressing an accident in which people died.

This set of words was selected because they are frequently used to define aircraft accidents or effects of aircraft accidents. Information about accidents, as the date and the site of the accident, is likely to appear near these words.

From this set of words, the expert selected synonyms and hyponyms of these words in WordNet to complete the set of context keywords. The expanded set of context keywords used was the following:

> BUMP, CLASH, COLLIDE, CRASH, HIT, JAR, KNOCK, RAM, SHOCK, SLAM, STRIKE, DESCEND, DOWN, FALL, LAND, DISAPPEAR, LOSE, BLEW, BOMB, EXPLODE, FIRE, DIE, KILL, PERISH, DIVE, NOSEDIVE, PLUMMET, PLUNGE.

Finally, this set was further automatically extended by adding all forms of each word using a morphological tool, resulting in the following bag of words:

> BLEW, BLEW_UP, BLOW, BLOWING_UP, BLOWN, BLOWN_UP, BLOW_UP, BOMB, BOMBED, BOMBING, BOMBS, BUMP, BUMPED, BUMPING, BUMPS, CLASH, CLASHED, CLASHES, CLASHING, COLLIDE, COLLIDED, COLLIDES, COLLIDING, CRASH, CRASHED, CRASHED-INTO, CRASHES, CRASHING, DESCEND, DESCENDED, DESCENDING, DESCENDS, DIE, DIED, DIES, DISAPPEAR, DISAPPEARED, DISAPPEARING, DISAPPEARS, DIVE, DIVED, DIVES, DIVING, DOWN, DOWNED, DOWNING, DOWNS, DYING, EXPLODE, EXPLODED, EXPLODED-OVER, EXPLODES, EXPLODING, FALL, FALL-OUT, FALLEN, FALLING, FALLING, FALLS, FELL, FIRE, FIRED, FIRES, FIRING, HIT, HITED, HITS, HITTING, JAR, JARED, JARING, JARRED, JARRING, KILL, KILLED, KILLING, KILLS, KNOCK, KNOCKED, KNOCKED-DOWN, KNOCKING, KNOCKS, LAND, LANDED, LANDING, LANDS, LOSE, LOSING, LOST, NOSEDIVE, NOSEDIVED, NOSEDIVES, NOSEDIVING, PERISH, PERISHES, PERISHED, PERISHING, PLUMMET, PLUMMETED, PLUMMETING, PLUMMETS, PLUNGE, PLUNGED, PLUNGES, PLUNGING, RAM, RAMED, RAMING, RAMMING, RAMS, SHOCK, SHOCKED, SHOCKING, SHOCKS, SLAM, SLAMED, SLAMING, SLAMMED, SLAMMING, SLAMS, SMASH, SMASHED, SMASHES, SMASHING, STRIKE, STRIKED, STRIKES, STRIKING, GO-DOWN, GOES-DOWN, GOING-DOWN, WENT-DOWN, GONE-DOWN.[1]

[1] Note that not all words returned by the morphological tool are correct words in English. The morphological tool tries to find all possible derivations for one word by following very simple rules. This fact does not affect the system performance because incorrect derivations will not be present in the corpus.

The set of context keywords for *noun and preposition groups* (that is, noun context words) was defined in the same way by selecting an initial set of noun words, expanding it with hypernyms and hyponyms in WordNet of these words, and finally by expanding the set of noun words applying a morphological tool. The final set of noun context keywords selected for `CRASH-SITE` and `CRASH-DATE` slots was:

> ATTACK, ATTACKS, BOMBING, BOMBINGS, DEATH, DEATHS, DESTRUCTION, DISASTER, DISASTERS, DOWN, EXPLOSION, EXPLOSIONS, FALL, FALLING, FALLS, TERRORISM, TRAGEDY, CRASH, CRASHING, CRASHES, WRECK, WRECKING, WRECKS, COLLISION, COLLISIONS, COLLAPSE, COLLAPSES, SMASH, SMASHING, SMASHES, HIT, HITS, HITTING, STRIKING, CRASH-INVESTIGATION, CRASH-PROOF, CRASH-RELATED, SMASHED, STRIKINGLY, WRECKAGE, WRECKED, LOST.

Extracting synsets

In addition to the set of context keywords, the ESSENCE methodology requires when possible, for each slot, the definition of a set of semantic tags in WordNet (synset numbers) representing the kind of information to be extracted for that slot. This set of semantic tags, named *extracting synsets* in the methodology, allows the system to focalize and speed up the search of candidate sentences for building IE patterns. For the `CRASH-SITE` slot, were defined the following *noun* synsets in WordNet 1.6 that describe possible locations for a crash: 6666185, 6359477, 6668569, 2661119, 6691504, 6667942. These synset numbers correspond to:

6666185 meaning: body_of_water, water [the part of the earth's surface covered with water; "they invaded our territorial waters"]

6359477 meaning: region [a large indefinite location on the surface of the Earth; "penguins inhabit the polar regions"]

6668569 meaning: land, ground, soil [what plants grow in (especially with reference to its quality or use); "the land had never been plowed"; "good agricultural soil"]

2661119 meaning: facility, installation [something created to provide a particular service; "the assembly plant is an enormous facility"]

6691504 meaning: geological_formation, geology, formation [the geological features of the earth]

6667942 meaning: land, dry_land, earth, ground, solid_ground, terra_firma [the solid part of the earth's surface; "the plane turned away from the sea and moved back over land"; "the earth shook for several minutes"; "he dropped the logs on the ground"]

This set of semantic tags were selected by the expert according to the constraints expressed in the MUC task definition for filling this slot.

In other cases, the information to be extracted cannot be characterized using WordNet, because it is expressed in domain dependent words not covered by WordNet or because their specific structure. The CRASH-DATE slot is one example of the later case. Since WordNet is not able to identify its semantic structure in texts, we used a modified version of the syntactic analyzer MARMOT (as described in section 3.3) that is able to perform detection of dates in texts.

5.2.2 Flight-information Slots

Context keywords

In the same way that was done for crash-information slots, a set of context keywords for the DESTINATION and DEPARTURE was defined. The initial set of context keywords contained words describing actions like traveling, leaving an airport and approaching to an airport. The set of verb context keywords was:

> LEAVE, GO, RETURN, TRAVEL, LAND, FLY, APPROACH, VISIT, ARRIVE, LAUNCH, BOUND.

while the initial set of noun context keywords for the same slots was:

> APPROACH, DESTINATION, FLIGHT, TRAVEL, LANDING, ARRIVAL, VISIT, DEPARTURE, LIFTOFF.

These sets of words were further completed with synonyms and hyponyms in WordNet as it was done in the previous case.

For the AIRLINE and MANUFACTURER slots, the set of seed context keywords referred to buy and sell (general activities that are related to companies) and to operate and build (specific activities of this kind of companies). The set of verbs was:

> BUY, ORDER, OWN, RENT, BORROW, DELIVER, OPERATE, BUILD.

while the initial set of noun context keywords for the same slots was:

BUILDER, RENTER, OWNER.

These sets of words were further extended with the help of WordNet and the morphological tool as in the previous cases. Then, these sets of context keywords were completed by adding the context keywords used for crash information slots. The reason for including also the set of context keywords defined for crash information slots is that when describing a crash it is usual to refer to the airline or to the manufacturer of the plane.

Finally, context keywords for the AIRCRAFT slot included all context keywords of the other slots, that is, the union of context keywords for CRASH-SITE, CRASH-DATE, DESTINATION, DEPARTURE, AIRLINE and MANUFACTURER slots.

Extracting synsets

The semantic information to be extracted for DESTINATION and DEPARTURE slots are physical places. The expert selected synsets in WordNet to cover geographic regions (including countries and cities) and facilities to cover airports and other installations. The obtained set is a subset of the extracting synsets for the CRASH-SITE slot (because destination and departure locations are usual places of a crash), that does not include generic places as, for example, oceans. The final set of extracting synsets defined by the expert was:

6359477 meaning: region [a large indefinite location on the surface of the Earth; “penguins inhabit the polar regions”]

6667942 meaning: land, dry_land, earth, ground, solid_ground, terra_firma [the solid part of the earth’s surface; “the plane turned away from the sea and moved back over land”; “the earth shook for several minutes”; “he dropped the logs on the ground”]

2661119 meaning: facility, installation [something created to provide a particular service; “the assembly plant is an enormous facility”]

The definition of the extracting synsets for the AIRCRAFT slot was more difficult. In this case, WordNet provides some covering because it includes the generic concept of aircraft. Then the expert firstly included the following WordNet synsets to the set of extracting synsets for the slot:

2170808 meaning: aircraft – (a vehicle that can fly)

3215456 meaning; reaction-propulsion engine, reaction engine – (a jet or rocket engine based on a form of aerodynamic propulsion in which the vehicle emits a high-speed stream)

Nevertheless, there are also other terms referring to planes that are not covered by WordNet. For instance "F14" is an example of such terms that are not covered by WordNet. To solve this problem, the expert used a list of airplane models in order to supply the lack of covering of WordNet. In the semantic analysis of sentences (see section 3.3), when one word was not recognized by WordNet, the system checked whether the word was in the list of airplane models or not. If the word was in the list, it was marked with the special tag `@PLANE@` which, at the same time, was added to the set of extracting synsets for the `AIRCRAFT` slot.

We encountered a similar problem when defining the extracting synsets for the `MANUFACTURER` slot, because plane manufacturers are organizations that are not covered by WordNet. We had to add a list of companies and acronyms of companies that is used in the semantic analysis step to tag words uncovered by WordNet with the special tag `@ORG@` which, at the same time, was used as the extracting synset for the `MANUFACTURER` slot.

The definition of the extracting synsets for the `AIRLINE` slot was as follows. As in the case of manufacturers, airlines are organizations and thus the semantic tag `@ORG@` just described was also defined as an extracting synset for this slot. But following the MUC guidelines, the `AIRLINE` slot can also be filled with proper names of persons or families when the owner of the crashed aircraft is an individual. For identifying these cases, we also added to MARMOT a hand made entity recognizer for person names, that tags them with the special tag `@PER@` which was also used as an extracting synset for the `AIRLINE` slot.

5.2.3 Other Parameters

The definition of the context keywords and the extracting synsets is done at the task definition step of the methodology (see section 3.1). Other steps of the method require values for some parameters to be selected. In our experiments we have selected the following values:

Window width

In the windowing step of the method (see section 3.4.2), the influence of a context keyword in the text is limited to a number of syntactic groups surrounding it. The

window width parameter determines how many syntactic groups placed before and after the context keyword would be considered in order to generalize patterns and to extract information from them. The effect of this parameter is rather obvious: the widest the window the higher recall but the lesser precision. In the experiments we selected heuristically a window width of 5 (that is, two syntactic groups before and two syntactic groups after the context keyword). Section 6.3 will report experimental results for one slot using other window width.

The ELA parameters

The ELA learning algorithm depends on the following parameters: θ_1, θ_2 and, either w_1 and w_2 or $|M|$.

The first parameter, θ_1, determines how high we can climb in the WordNet hierarchy to allow the matching of patterns. In WordNet, all nouns are hyponyms of the root object and thus, all objects do match. We put a limit in order to avoid so general matchings. We found θ_1=3 an adequate value for our experiments, that is, we did not allow matchings in the three first levels of the WordNet hierarchy of concepts.

The second parameter, θ_2, puts a limit on how much generalization is allowed in a single generalization step. This is expressed as the maximum number of levels in the WordNet hierarchy that are allowed to be climbed in order to find a matching concept. We tested different values from 2 to 15 and we found that values in the range 8 to 13 worked well. We decided set θ_2=10 in our experiments.

Finally, ELA considers the cost of any generalization step as the sum of the cost of the minimum generalization necessary to allow the matching of groups which would remain in the pattern, plus the cost of the generalization made in order to remove the no matching groups (see in page 47 formula 4.1). The w_2 parameter weights the former while the w_1 parameter weights the later. An special case is when the user prefers to consider patterns with a constant number of matching groups (see end of section 4.1. This corresponds to the case when the user begins learning with specific patterns having a fixed number of matching groups, $|M|$, and sets the parameter w_1 to ∞. In this way we do not allow any group to be removed from the pattern. Since the initial number of matching groups is $|M|$, it will remain constant through the generalization process. We chose for these experiments to set the initial number of matching groups $|M|$ to 2 (that is, each pattern will contain 2 matching groups: one group for the context keyword and another for one extracting synset).

We conducted preliminary experiments in order to guess the adequate number of syntactic groups for the patterns to be learned. We also tried to learn

patterns with 3 groups (that is, $|M| = 3$). Results for these experiments showed that while precision of patterns was higher than patterns with 2 groups, recall dropped too much. The final F measure results showed that patterns with 2 matching groups were more suitable to the proposed domain when given the same relevance to recall and precision (that is, when considering the F measure with $\beta = 1$) than patterns with 3 matching groups.

Minimum number of cases covered by an IE pattern

Finally, in order to ensure that the patterns generated by ELA are minimally generalized, we require that each generated pattern could be applied at least in 3 cases in the training corpus. This constraint prevents the generation of too specific patterns which are applicable only to one or two sentences. Consequently, it reduces the number of learned patterns by ensuring that the patterns have a minimum applicability.

This constraint is implemented in the *learn-one-pattern* function described in page 53. At the end of the function, positive evaluation of the best pattern requires that the pattern covers at least 3 cases in the training test.

5.3 Results in a MUC-like Evaluation

From the sets of context keywords, the sets of extracting synsets and the values for parameters described in the previous section, the ESSENCE method was applied to the 100 texts that compose the training set for the task. The set of training and test texts is fixed and delivered by the MUC organizers.

As it is indicated in section 4.2, the function *learn-one-pattern* obtains a list of patterns sorted by their generalization degree, being the last pattern the most general one. From this list, it must be selected the best pattern which is returned by the function. The selection of the best pattern can be done either by the expert or automatically by using simple heuristics.

In the MUC-like experiments reported in this chapter we selected the best pattern by showing to the user the sentences in the training texts that the most general pattern is able to cover. This set of sentences has not to be calculated because it is exactly the set of sentences that generated the pattern. We simply store the identifier of the sentences covered by each pattern as part of the information the pattern holds while ELA is building it. The most general pattern (that is, the last one of the generalization list) is applied to these sentences showing to the user the information extracted by the pattern. The extracted values for the first 40 sentences (at the maximum) are shown to the expert. The expert tags

each extracted value as correct or incorrect for the slot intended to be filled. The sampling of 40 sentences is used to evaluate the *estimated precision* for each pattern in the list. This estimation is defined as the amount of correctly extracted information out of the total amount of information extracted by the pattern, that is:

$$EstimatedP_i = \frac{ExtractedTaggedRight_i}{TotalExtracted_i} \tag{5.1}$$

where i stands for the i-nth pattern in the list.

In addition, we define the *relative recall* for one pattern as the amount of correctly extracted information by the pattern compared with the amount of correctly extracted information by the most general pattern.

$$RelativeR_i = \frac{ExtractedTaggedRight_i}{ExtractedTaggedRight_n} \tag{5.2}$$

where n is the number of the last pattern of the generalization list (the more general one). Notice that $RelativeR_i$ values are in the [0..1] range because the amount of information extracted by the most general pattern is always higher than the information extracted by any other pattern in the list of patterns. Note that once the expert has tagged the information extracted by patterns as correct or incorrect for estimating the precision of patterns, the computation of the relative recall is made without any additional intervention of the expert.

With these definitions on mind we guess the best pattern by following the *best relative F measure* criterium. The best pattern is the one with highest *Relative F measure*, defined as:

$$RelativeF_i = \frac{2 \cdot RelativeR_i \cdot EstimatedP_i}{RelativeR_i + EstimatedP_i} \tag{5.3}$$

where i stands for the i-nth pattern in the list ($1 \leq i \leq n$). This is the formula to calculate the F measure with $\beta = 1$ but replacing recall and precision by relative recall and estimated precision, respectively.

The set of patterns obtained by applying ESSENCE to the training texts with the procedure described above to select the best pattern, was tested on both the training set and the test set[2]. Recall, precision and F measure results for each slot to be filled in the IE task are shown in table 5.1.

[2]The goal of testing the patterns in the training set was to test whether those patterns were representative for the training texts, not for validating them.

	Training set			Test set		
Concept	**R**	**P**	**F**	**R**	**P**	**F**
Crash information						
Crash Site	67.6	68.0	**67.8**	59.4	51.2	**55.0**
Crash Date	78.3	62.0	**69.2**	75.4	82.6	**78.8**
Flight information						
Aircraft	69.0	100.0	**81.6**	65.0	100.0	**78.8**
Airline	55.7	57.9	**56.8**	65.2	55.2	**59.9**
Manufacturer	56.2	53.6	**54.9**	39.5	62.3	**48.4**
Departure	52.1	87.7	**65.3**	57.6	51.5	**54.4**
Destination	54.3	94.8	**69.1**	60.7	72.9	**66.3**

Table 5.1: Results for the Aircraft Crash domain.

Results show an average level for F measure of 66.4% in training and 63.1% in testing, which are reasonably high compared to the overall performance of other systems in similar MUC tasks (MUC 1991; MUC 1992; MUC 1993; MUC 1995; MUC 1998).

Note that for some slots the results obtained on the test set are better than those obtained on the training set. This effect appears because of the small number of texts in both sets. It is also worth to note the high differences between results on the training set and results on the test set, technically known as *high variance*, that suggest that these results are not a definitive validation of the proposed method. In the next chapter we will describe a more complete set of experiments that average results for different runs on different training and test sets to reduce variance produced by a single test.

Some of the results shown in table 5.1 could be improved upon. The worse results are for the `MANUFACTURER` and `AIRLINE` slots, in both training and test sets. In the description of the *Semantic Tagging* module in section 3.4.3, we suggested that the information to be extracted from a syntactic group is usually the type of information in its *headword*, but this is not always true. For example, the `MANUFACTURER` information usually takes the role of a *modifier* of the crashed plane which is the headword of the group, for example “Boeing 747”. This explains the poor recall values for this slot. The scores for the `AIRLINE` and `MANUFACTURER` slots could be greatly improved by also allowing the extraction of information from *modifiers* of headwords.

The procedure for extracting information from modifiers is straightforward: when a specific type of target information is often used as a modifier of a semantic class of headwords (this is easy to guess for the expert who defines the task), then the expert has to also define specific extracting synsets for this kind of headwords. Then, when the ESSENCE method is applied, IE patterns for these headwords will also be learned. Later, in the extraction phase, these patterns are applied as usual but when they extract a syntactic group the returned information to fill the slot will not be the headword but the modifier.

For example, airlines are very often used as modifiers of aircrafts (for example, "TWA 747", etc.). In order to extract the airline information when it plays the role of modifier of a plane, the user defines the *extracting synsets for a modified head* with the semantic tags for aircraft models (that is, the extracting synsets for the AIRCRAFT slot that we have seen above). Later, ELA will learn patterns for extracting aircraft models among the patterns for extracting the AIRLINE slot. These are special patterns in the sense that they do not extract information carried by the headword but by a modifier of the headword. When one of these patterns extracts a noun or preposition group with the headword semantically tagged as @PLANE@, the system will search in the syntactic group a modifier semantically tagged as @ORG@ or @PER@ (the real extracting synsets for AIRLINE), and use this information to fill the AIRLINE slot. When the system does not find a modifier with these tags, nothing is extracted. Note that in the latter case, the application of the pattern is not counted for recall or precision scores because nothing is extracted.

The reason for not generating IE patterns for modifiers in the same way as for headwords, i.e., defining extracting synsets for the target modifiers, is that they cannot be generalized using WordNet. There is no hyponym/hypernym relation between modifiers. If we wish to acquire IE patterns taking advantage of generalization using the WordNet hierarchy, we can only rely on generalizing the headword.

In the next chapter we will describe a set of experiments conducted to test the procedure for extracting information from modifiers. We applied this procedure to extract information for filling the AIRLINE and MANUFACTURER slots. Results indicate that recall for both slots can achieve up to 75%.

Chapter 6

Detailed Empirical Study

This chapter reports the results obtained applying ESSENCE on the same domain as that of the previous chapter but with an evaluation procedure designed to obtain a more sound and complete evaluation of the learning abilities of ESSENCE than the obtained by following MUC rules. In particular, we conducted an exhaustive set of experiments to evaluate the performance of the system with different number of training texts and with different values for some parameters of the learning algorithm.

6.1 Evaluation Procedure

One problem with testing in MUC-style competitions is that training and testing of the systems are performed on exactly one fixed training set and one fixed test set. Although dealing with one fixed training set and one fixed test set is the usual procedure when building an IE system (that MUC tries to evaluate), this method does not allow a fair comparison between learning systems.

It is well known that in order to compare different learning systems, them must be compared by their performance on different training and test sets. Randomness in the selection of one training set and one test set could produce by chance a non representative training set or a extremely difficult (or simple) test set. The average of the results for different training and test sets reduce the misleading effects of the random selection of texts.

In order to soundly evaluate the learning ability of ESSENCE on the data set described in the previous chapter, we decided to average the results of learning on 20 random partitions of the whole data set into a training set and a test set.

The size of the test set was 20 texts, that is relatively small compared with the test set in MUC competitions (100 texts), but since we performed 20 tests (one

for each partition), the total number of tests performed was 20x20 = 400 which is statistically more significant than MUC tests.

The training set was composed of the remaining 180 texts from the data set. Since one of the features of ESSENCE we wanted to know is the evolution of its performance as the number of training texts increases, we have divided the training set into five subsets containing 20, 60, 100, 140 and 180 texts, respectively. We named these subsets Trn_{20}, Trn_{60}, Trn_{100}, Trn_{140} and Trn_{180} respectively. These subsets fulfill that $Trn_i \subset Trn_j$ when $i < j$, that is, training set Trn_{20} is composed of 20 texts of the training set, these 20 texts and 40 more texts compose Trn_{60} set, all them and new 40 texts compose Trn_{100}, and so on until the whole data training set is contained in Trn_{180}. The details of the data preparation procedure are described in algorithm 3.

There are other ways to evaluate the performance of learning systems, the most usual are *k-fold cross validation* and *leave one out*. *k-fold cross validation* consists in dividing the data set into k disjoint sets, each one with the same number of texts (k/n texts, where n is the number of texts in the whole data set). Validation is done by averaging results for k evaluations where at each evaluation a different subset is used as the test set while the remaining are used for training. Thus, validation consists in the average of k runs on test sets composed of k/n texts. When k is 1, we have the *leave one out* procedure.

The testing procedure we propose is more accurate than 10-fold cross validation because we perform 20 evaluations on test sets with 20 texts instead of 10 tests on test sets with 20 texts. It is also more accurate than 20-fold cross-validation because the later consists on 20 evaluations on test set with only 10 texts. Nevertheless, the procedure we propose has the drawback that the 20 test sets are not disjoint an thus, results for different runs are slightly correlated (they are not statistically independent).

With the testing procedure we propose, we are interested in studying the number of learned patterns and the precision, recall and F measures obtained depending on the number of texts supplied for training. We expected that as the number of training texts is increased, recall will also be increased. In addition, one of the goals of ESSENCE is the ability to learn from a relative small set of texts. This hypothesis will be tested by inspecting how the learning measures evolve as the number of training texts is increased.

As we stated in section 4.2, the *learn-one-pattern* function used by ELA needs a procedure to choose the best pattern from the list of generalized patterns. We described in the same section a procedure for selecting the best pattern (that has been used in the MUC like experiment - see section 5.3). This procedure tries to balance estimations of recall and precision to select the best pattern.

Algorithm 3 Data preparation procedure

Require: DS (Data set with 200 texts)

{Build 20 random divisions of the whole data set into training set Trn (180 texts) and test set Tst (20 texts). Also divide each training set Trn into sets with 20, 60 ,100, 140 and 180 texts.}

for k = 1 to 20 **do**
  $Tst^k \leftarrow$ RandomSampling(DS, 20) { *k-th* Test set}
  $Trn^k \leftarrow$ DS $- Tst^k$ { *k-th* Training set}
  $i \leftarrow 20$
  $Trn^k_i \leftarrow$ RandomSampling(Trn^k, 20)
  while $i \leq 180$ **do**
    $Trn^k \leftarrow Trn^k - Trn^k_i$
    $Trn^k_{i+40} \leftarrow Trn^k_i \cup$ RandomSampling(Trn^k, 40)
    $i \leftarrow i$ + 40
  end while
end for

Another procedure for selecting the pattern from the list of generalized patterns is to sort the list by their precision in a sample of the training set (*estimated precision* as it is described in section 5.3) and *choosing the first pattern with estimated precision higher than a minimum value* which we will see as a threshold. Since precision is inversely proportional to recall, the minimum allowed value of estimated precision for IE patterns can be seen as a tuning parameter of the learning algorithm that allows the user to bias the learning either towards high recall or high precision. A low threshold of estimated precision biases learning towards high recall of patterns, while a high threshold of estimated precision biases learning towards precise patterns.

Taking this into account, we will also study in this chapter how the selection of patterns by requiring different values for the threshold of estimated precision influences recall, precision and F measures on the test set.

Finally, the tradeoff between quality (precision) and covering (recall) is another interesting property of learning algorithms that will be studied by showing actual recall-precision plots.

With the training and test sets obtained in the way described above, ESSENCE will be tested as shown in algorithm 4 in order to obtain the number of patterns

Algorithm 4 Test procedure

Require: Set of training and test sets obtained in algorithm 3

{20 runs test procedure for each slot considering the number of texts and requiring a minimum estimated precision of patterns}

for each slot s **do**
  for each number of training texts value i in {20, 60, 100, 140, 180} **do**
    for k = 1 to 20 **do**
      for each relative precision value p in {0, 10, 20, 30, ..., 90, 100} **do**
        Set $IE^k_{i,p}(s)$ to the set of patterns for slot s obtained with ESSENCE from Trn^k_i with relative precision higher or equal to p
        Set $R^k_{i,p}(s)$ to the Recall results of testing Tst^k with $IE^k_{i,p}$
        Set $P^k_{i,p}(s)$ to the Precision results of testing Tst^k with $IE^k_{i,p}$
        Set $F^k_{i,p}(s)$ to the F measure results of testing Tst^k with $IE^k_{i,p}$
      end for
    end for
    Print average results of $|IE^k_{i,p}(s)|$, $R^k_{i,p}(s)$, $P^k_{i,p}(s)$ and $F^k_{i,p}(s)$ for k varying from 1 to 20
  end for
end for

learned, and recall, precision and F measures for the chosen domain.

6.2 Results

In this section we report and analyze the results obtained by the ESSENCE method applying the evaluation procedure described above. The experiments were based on the air crashes domain described in the previous chapter. The sets of context keywords and extracting synsets for each slot are the same than we defined in the MUC-like experiment. The values for the parameters of the ESSENCE method also remain unchanged.

The only difference with the MUC-like experiments (in addition to the evaluation method) is that now we have implemented the procedure that allow the system to extract information from modifiers. We tested the new procedure to extract information for the `AIRLINE` and `MANUFACTURER` slots.

In the case of the `MANUFACTURER` slot, since manufacturers are often used

as modifiers of aircrafts (for example, “Boeing 747”), the expert defined as extracting synsets for a modified head the extracting synsets for the AIRCRAFT slot. Thus, patterns to identify aircraft models will be learned in order to extract manufacturer information from their modifiers.

Finally, since airlines are also often used as modifiers of aircrafts (for example, “TWA 747”), the expert defined as extracting synsets for a modified head the extracting synsets for the AIRCRAFT slot. In addition, since airlines are also modifiers of the word “flight”, for instance “TWA flight”, the user also defined as extracting synsets for a modified head the WordNet synset:

195002 meaning: flight [a scheduled trip by plane between designated airports; “I took the noon flight to Chicago”]

In the following section we report the results obtained applying the evaluation procedure proposed. In particular we are interested in studying the number of learned patterns, recall and precision measures, and the relationship between recall and precision.

6.2.1 Number of Learned Patterns

One of the goals of ESSENCE is to learn IE patterns with a small number of training texts when compared with other unsupervised approaches for learning patterns. One way to measure the success of this goal is by analyzing the number of extraction patterns acquired as the number of texts for training increases. Figures 6.2 to 6.7 show for each slot of the task how many IE patterns are learned as the number of training texts increases. Each curve in a plot represents the number of learned patterns when a minimum allowed value of estimated precision is required.

All plots show that by increasing the threshold of estimated precision, the number of learned patterns drops. This was expected: sometimes the *learn-one-pattern* function of ELA generates list of patterns without any pattern with an estimated precision above the threshold. In this case no pattern is returned. As the minimum allowed value of estimated precision is increased this case is more likely to appear.

Plots also show that as the number of training texts is increased, the number of learned patterns also increases. This effect was expected too: the addition of new training texts allow the learner to discover new patterns in texts[1].

[1]Nevertheless, note that this general statement is not fulfilled for high values of threshold of estimated precision. For instance, in figure 6.1, for threshold of estimated precision 100% and training on a number of texts from 140 to 180, the number of learned patterns seems to

However, the more interesting information that these plots show is that the increase in the number of patterns is appreciable from 20 to 100 texts, but from 140 texts on, the increase in the number of patterns is not so large. The effect is more evident in sets of patterns for which a minimum value of estimated precision between 40 and 100 is required.

Thus, at first glance, the property we are trying to validate, the learning from a small set of training texts, is partially true (at least for sets of patterns for which a high threshold of estimated precision is required) for this domain and slots considering as small a number of training texts around 140.

Nevertheless notice that the fact that the number of patterns is increased does not imply that learning measures are will increased too, even for recall. Patterns that appear when training on 180 texts but do not appear when training on 140 texts, are patterns that usually have a *very low recall*. Notice that we needed 180 texts to discover them. Considering, in addition, that precision is independent from the number of texts used for learning, the resulting F measure for learning from 180 texts should not be increased significatively, even when the number of patterns grows. This explanation will be confirmed in the next sections where plots for recall and F measures are shown.

By inspecting plots from 6.2 to 6.7, we can find singularities that are worth to be commented.

In some cases, when we require a high minimum value of estimated precision, the system returns an empty set of patterns. This is the case for the slots `CRASH-DATE` (for threshold of estimated precision 100% and training texts from 100 to 140) and `MANUFACTURER` (for thresholds of estimated precision from 100% to 60% when learning from 20 texts, and for thresholds of estimated precision from 100% to 80% when learning from 60 texts). In the following sections, plots will not show information of recall and precision for this cases because the empty set of patterns has neither recall nor precision.

Another singular case is shown in the plot 6.6 for the `AIRCRAFT` slot. This plot shows only one curve. That means that whichever the minimum estimation precision we required, we obtained the same set of patterns. The cause of having only one curve is that all patterns had a 100% estimated precision. The high pre-

---

decrease. In fact, the number of patterns does not decrease, but that the number of patterns above a threshold of estimated precision decreases. Note that estimated precision is computed on a sampling of sentences *in the training set* which the pattern covers (see section 5.3). Thus, some patterns with estimated precision above a certain threshold, when are used on new texts, have a precision under that threshold. For instance, some patterns that had 100% estimated precision when tested on 140 texts, are not longer 100% reliable when the test considers 180 texts (the pattern fails at least in one case in the new texts). These are the patterns missing from 140 to 180 texts. This effect is more noticeable when requiring a high estimated precision threshold.

cision of these patterns is due to the specialized vocabulary for aircraft models. Every time we find a context keyword near a group with the head semantically tagged as @PLANE@, the headword is the model of the airplane that crashed. This did not happen in other slots for which we had to use a list of words to supply the lack of coverage of WordNet, like in the MANUFACTURER and AIRLINE slots for which we used the tag @ORG@. For instance, when extracting information by using patterns for the MANUFACTURER slot we extract something tagged as @ORG@, we are not sure that we have extracted a manufacturer. It could be an airline or another kind of organization. This partially explains why precision of patterns for MANUFACTURER and AIRLINE slots is not always 100%.

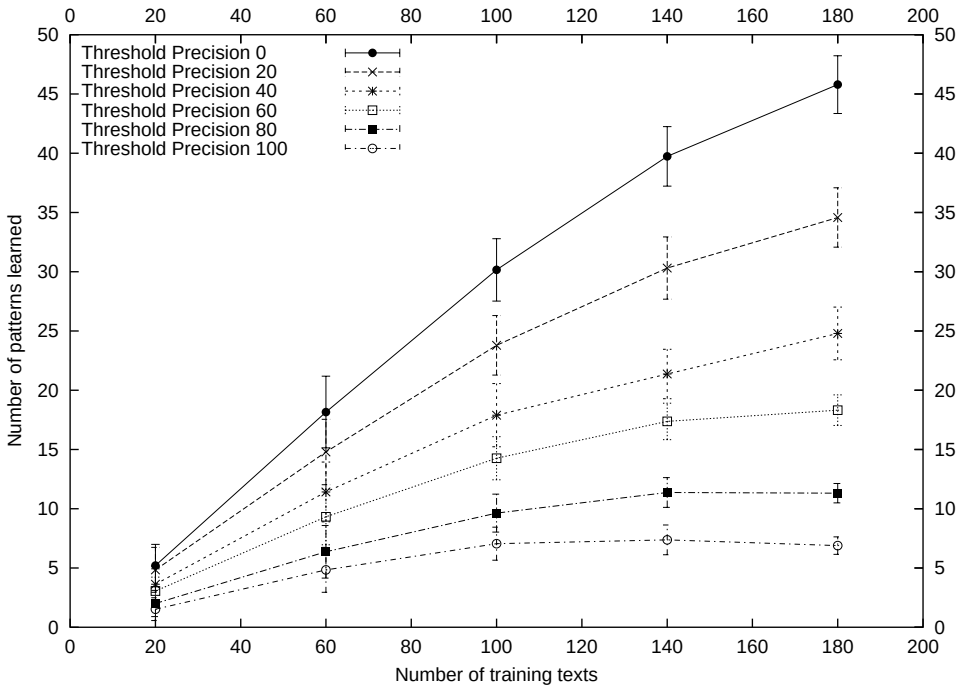


Figure 6.1: Evolution of the number of IE patterns learned above a threshold of estimated precision, for extracting CRASH-SITE information, as the number of training texts is increased. Margins show the standard deviation obtained for 20 runs.

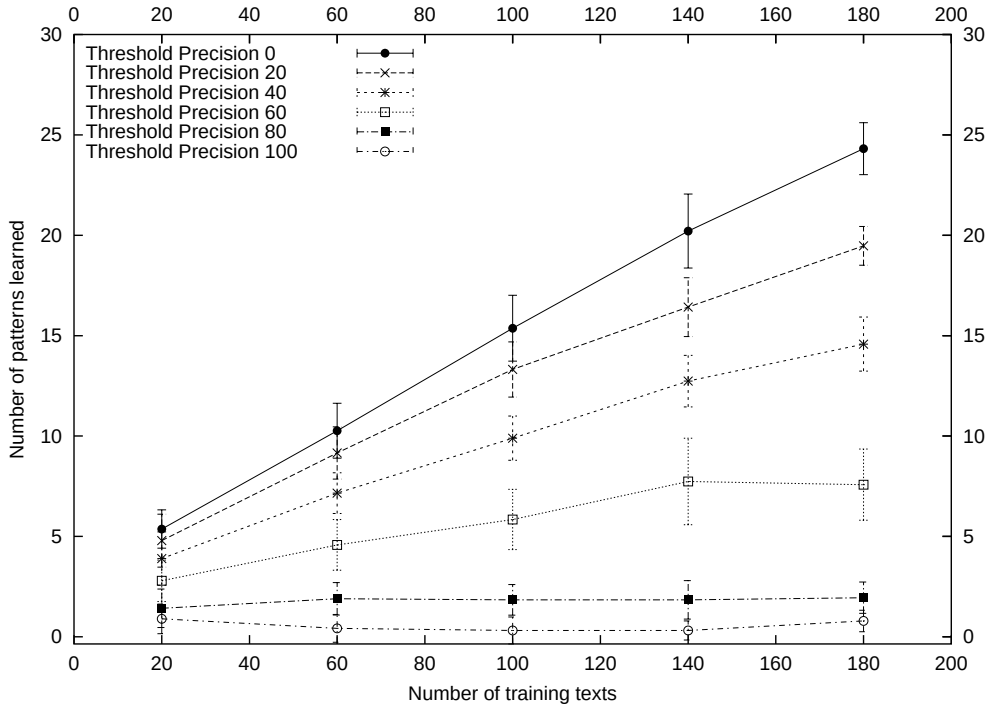


Figure 6.2: Evolution of the number of IE patterns learned above a threshold of estimated precision, for extracting CRASH-DATE information, as the number of training texts is increased. Margins show the standard deviation obtained for 20 runs.

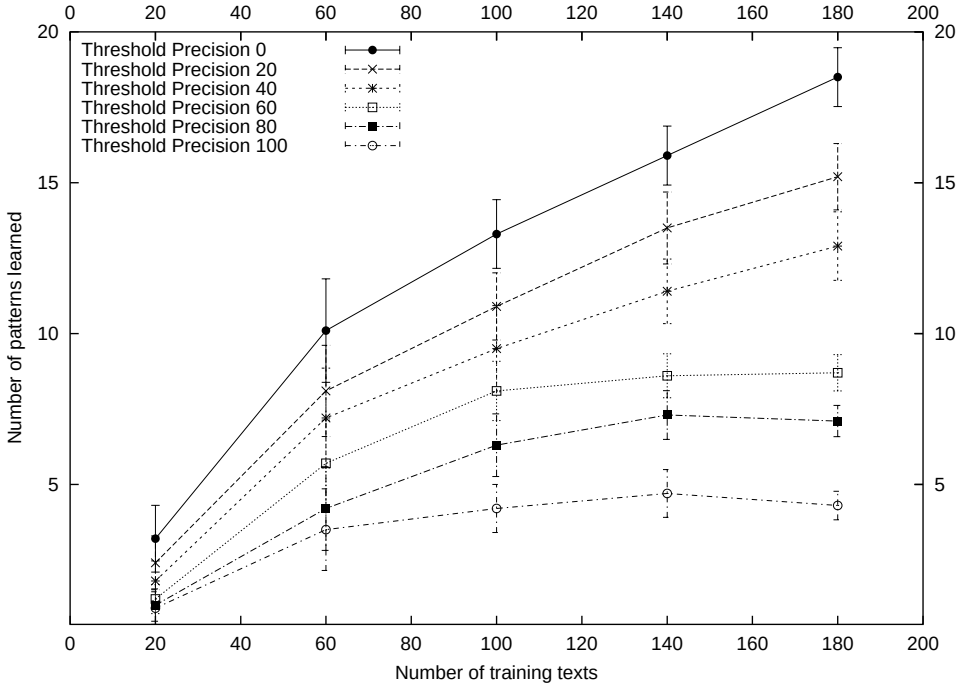


Figure 6.3: Evolution of the number of IE patterns learned above a threshold of estimated precision, for extracting DEPARTURE information, as the number of training texts grows. Margins show the standard deviation obtained for 20 runs.

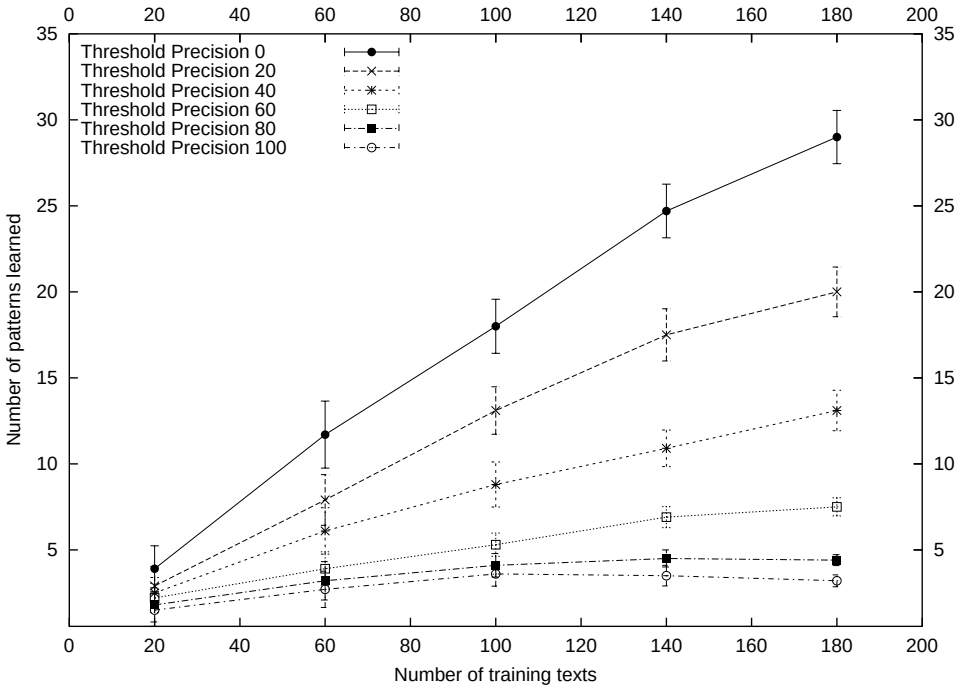


Figure 6.4: Evolution of the number of IE patterns learned above a threshold of estimated precision, for extracting `DESTINATION` information, as the number of training texts is increased. Margins show the standard deviation obtained for 20 runs.

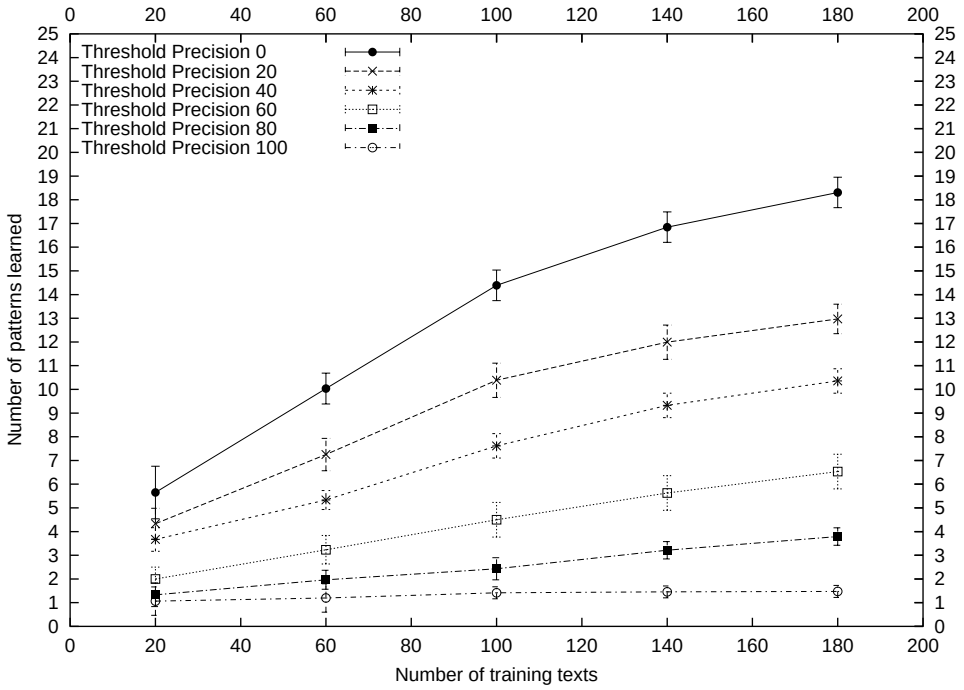


Figure 6.5: Evolution of the number of IE patterns learned above a threshold of estimated precision, for extracting `AIRLINE` information, as the number of training texts is increased. Margins show the standard deviation obtained for 20 runs.

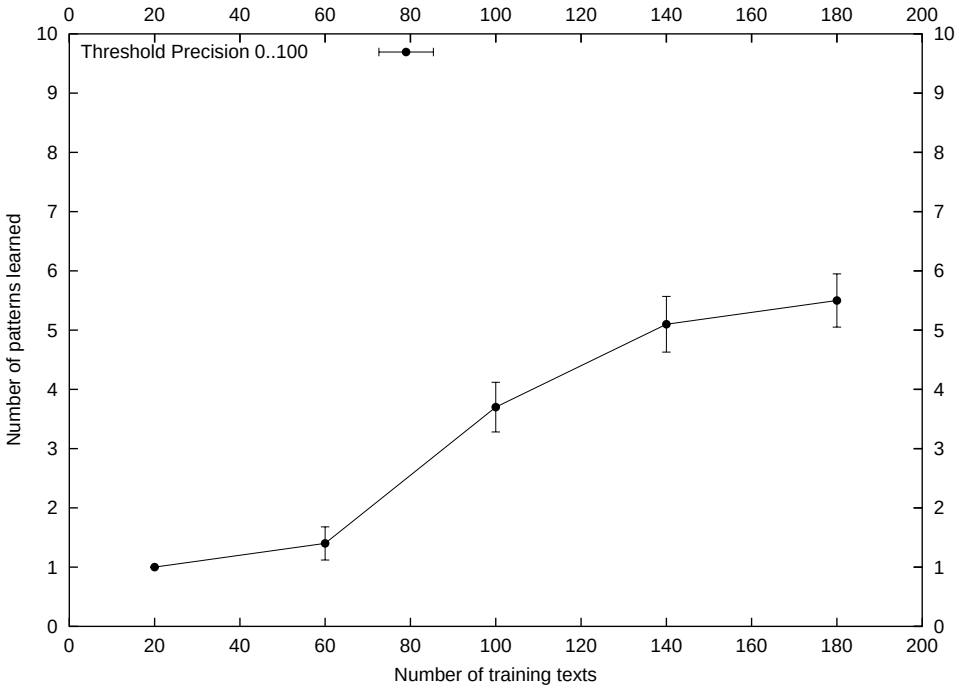


Figure 6.6: Evolution of the number of IE patterns learned above a threshold of estimated precision, for extracting AIRCRAFT information, as the number of training texts is increased. Margins show the standard deviation obtained for 20 runs.

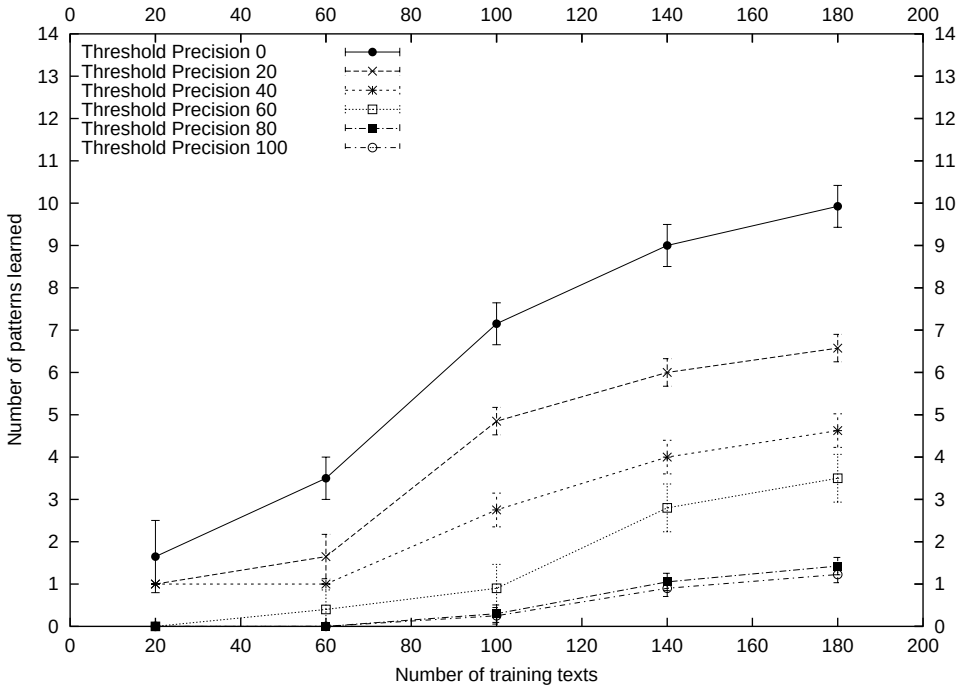


Figure 6.7: Evolution of the number of IE patterns learned above a threshold of estimated precision, for extracting the slot MANUFACTURER, as the number of training texts is increased. Margins show the standard deviation obtained for 20 runs.

6.2.2 Evolution of the Recall Measure

Another feature of ESSENCE interesting to know is how recall of the set of learned patterns evolves as the number of texts for training is increased and as the threshold of estimated precision changes.

Figures (a) from 6.8 to 6.14 show that recall do increase as the size of the training set is increased. The reason is that as the number of texts for training increases, the number of learned patterns increases too (see the previous section) and thus, recall also increases. Nevertheless, these figures also show that recall does not grow in the same way that the number of patterns. Note, for example, that in figure 6.8 (a) the difference in recall for curves with thresholds of estimated precision 0 and 20 is not proportional to the difference in the number of patterns shown in figure 6.1 for the same thresholds of estimated precision. The reason was explained in the previous section: patterns that appear late in the learning process have low recall, while patterns that appear early in the learning process have a high recall. Another reason can explain (at less extent) the difference between number of patterns and recall plots. When one information is extracted by two different patterns, index of recovered information is only increased once. Thus, some patterns are redundant and, though there are actually more patterns, they do not increase recall.

Another observation worth to mention is that the lower threshold of estimated precision, the higher recall.

Note that although, in general, recall increases with the number of training texts, sometimes it drops a little. For example, in figure 6.8 (a), in the curve with threshold of estimated precision 60%, recall decreases when training is increased from 140 to 180 texts. This fact is explained similarly to the case of the number of patterns. Some patterns learned from 140 texts had an estimated precision above 60% in the 140 training texts, but the same patterns disappear from the set of patterns for which a minimum value of estimated precision of 60% is required when learning is done with 180 texts, because their estimated precision on the training set is now below 60%. The loss of these patterns implies a loss of recall.

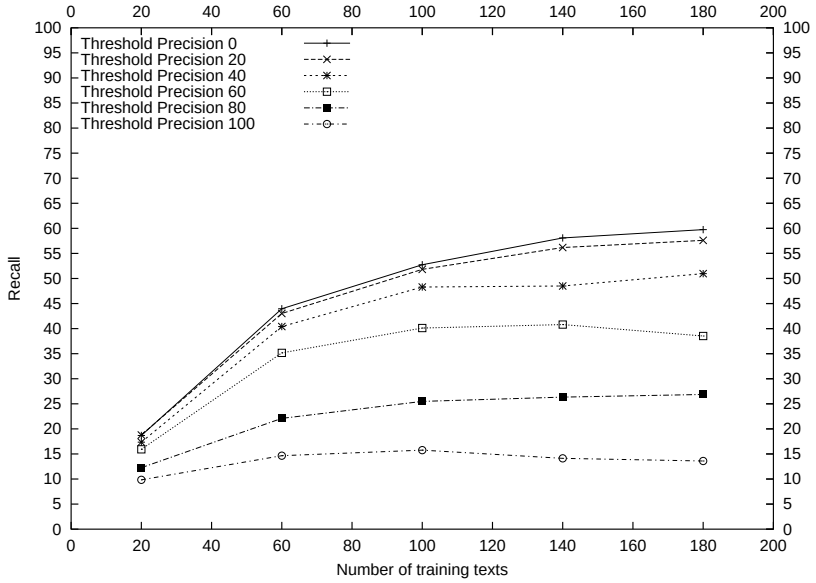
Figures (b) from 6.8 to 6.14 show the effect of the threshold of estimated precision on the recall measure. These figures show that differences in recall by learning from 140 texts and by learning from 180 texts are not significative whichever the threshold of estimated precision chosen. Thus learning seems to stop at 140 texts.

Notice also the following singularities:

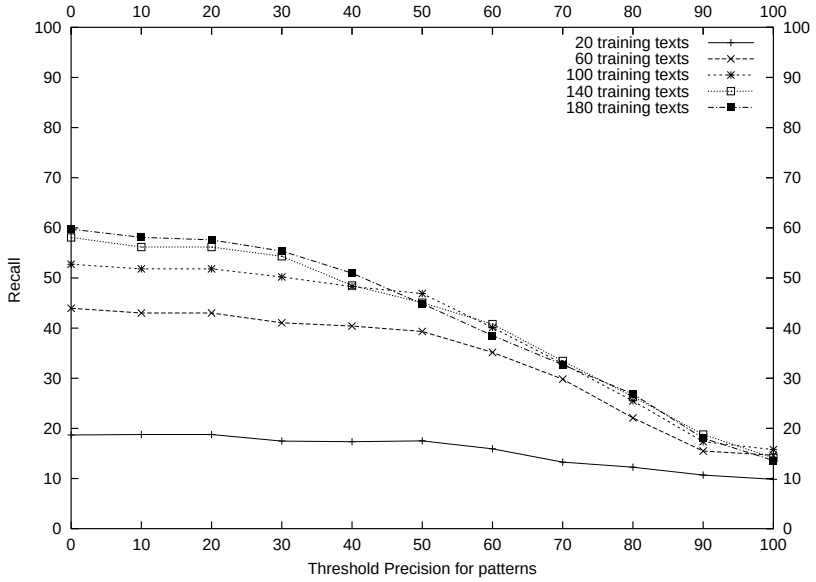
- First, recall scores for the AIRLINE and MANUFACTURER slots, are much higher than those reported for the MUC-like experiments. The explanation for this is that now we are using a procedure for extracting information from modifiers.

- Second, the curves in plots 6.9 (a) and 6.14 (a) for the CRASH-DATE and MANUFACTURER slots do not report some recall values for high thresholds of estimated precision, because the set of patterns returned by ESSENCE was empty (see previous section). In addition, for the CRASH-DATE slot at 100% of minimum allowed estimated precision and 60 training texts, we obtained a very low recall. This fact is explained because only 1 run out of the 20 runs performed, returned an unique pattern while the remaining runs returned the empty set. The average result for all the 20 runs is the value reported in the corresponding plot.

- Finally, in some cases only one pattern is responsible of a very high recall. This kind of patterns always appear very early in the learning process. They can be easily detected because for curves corresponding to similar threshold of estimated precision there is a big gap in between. For example, in figure 6.14 (a) for the MANUFACTURER slot there is a big gap between curves for 60 and 40 threshold of estimated precision. The gap is present since starting learning from 20 texts. The responsible is a pattern with estimated precision on the training text between 40% and 50% that is able to recover 38% of manufacturers. The other curves for threshold of estimated precision higher than 50 do not achieve this recall even when learning from 180 texts because the real precision of the pattern is below the minimum required precision of 50%.

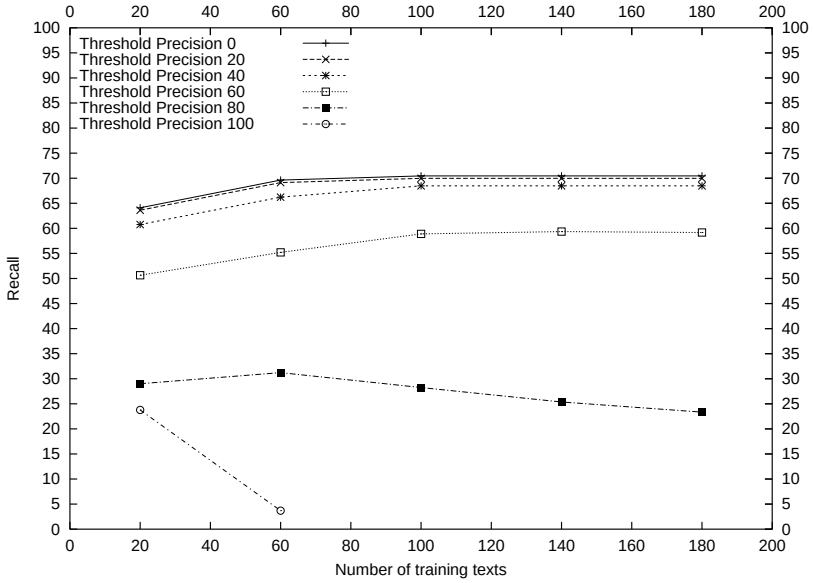


(a)

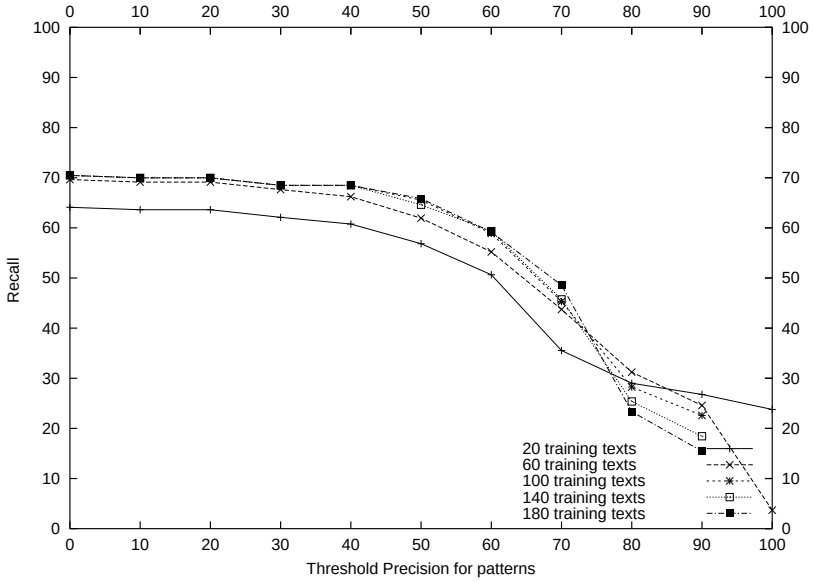


(b)

Figure 6.8: (a) Evolution of recall on test set for patterns learned with estimated precision above a given threshold, for extracting `CRASH-SITE` information, as the number of training texts is increased, and (b) evolution of recall on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

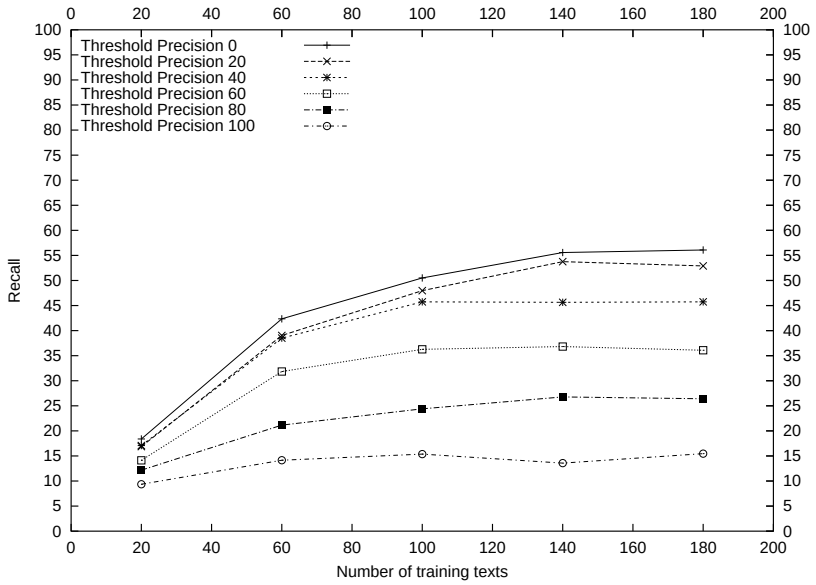


(a)

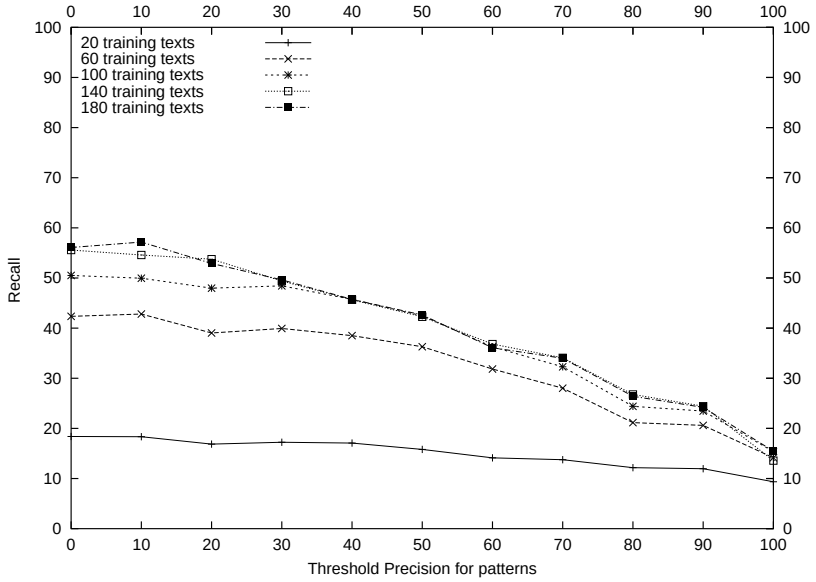


(b)

Figure 6.9: (a) Evolution of recall on test set for patterns learned with estimated precision above a given threshold, for extracting CRASH-DATE information, as the number of training texts is increased, and (b) evolution of recall on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

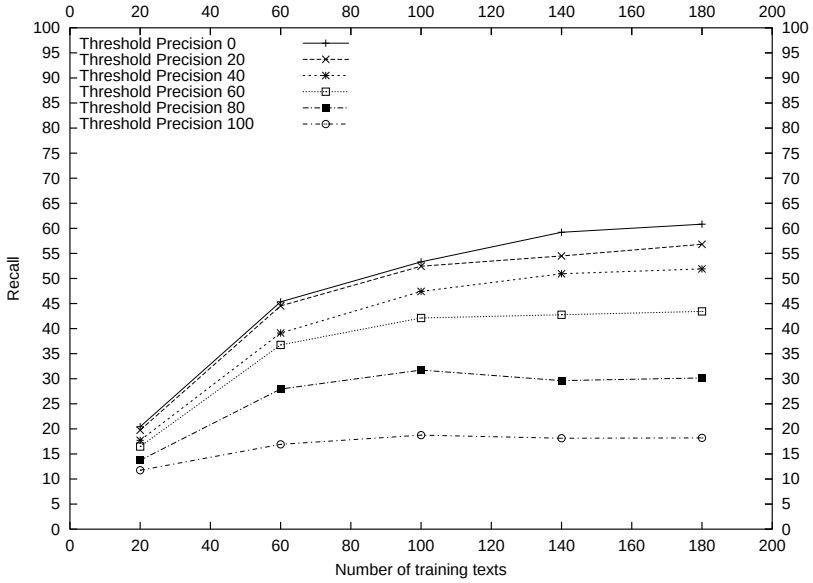


(a)

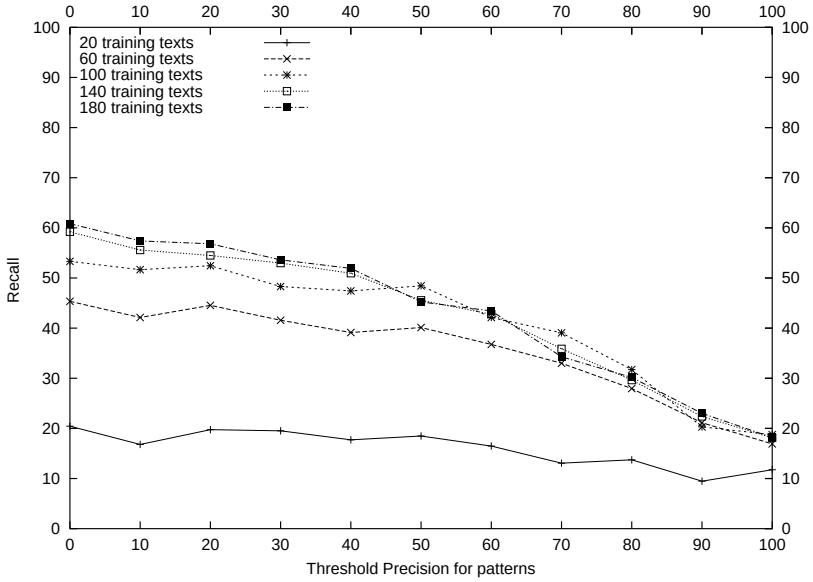


(b)

Figure 6.10: (a) Evolution of recall on test set for patterns learned with estimated precision above a given threshold, for extracting DEPARTURE information, as the number of training texts is increased, and (b) evolution of recall on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

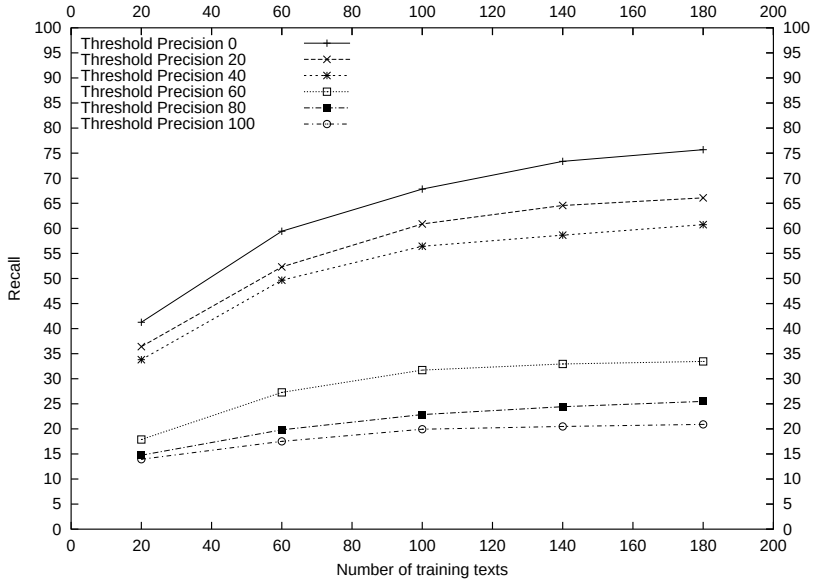


(a)

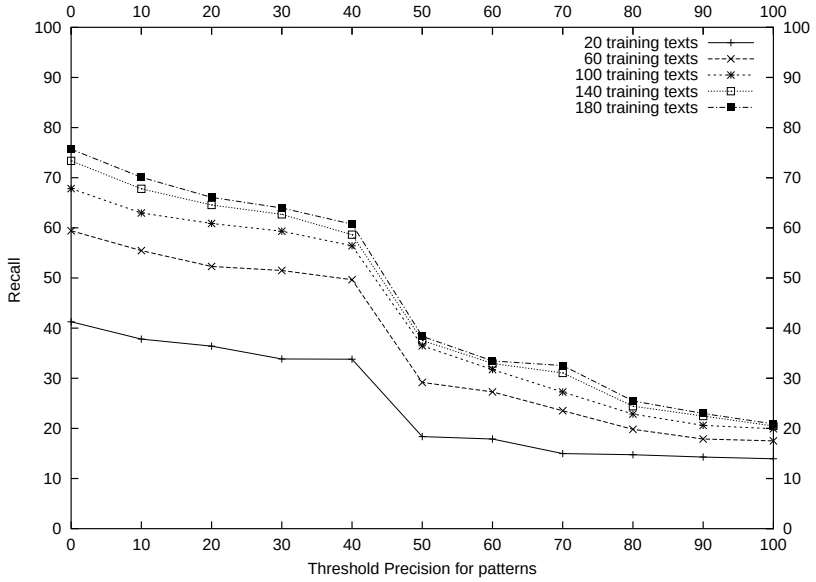


(b)

Figure 6.11: (a) Evolution of recall on test set for patterns learned with estimated precision above a given threshold, for extracting `DESTINATION` information, as the number of training texts is increased, and (b) evolution of recall on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

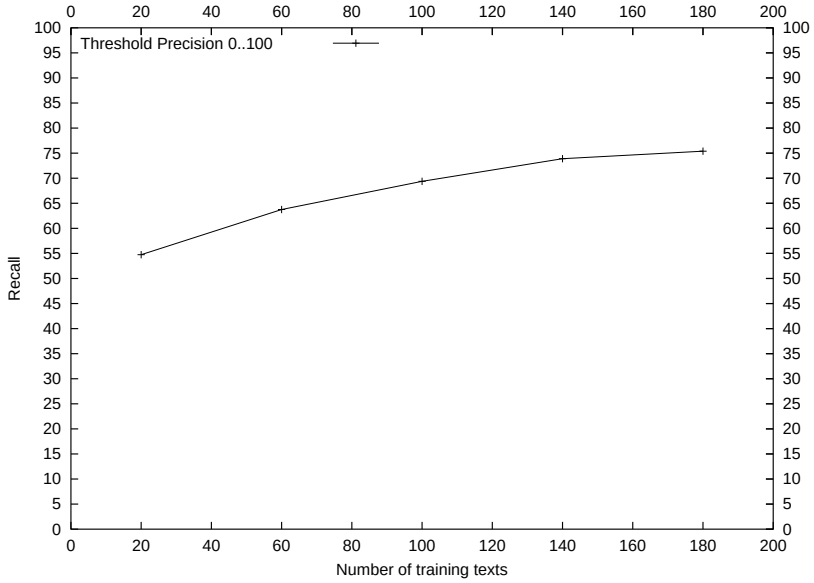


(a)

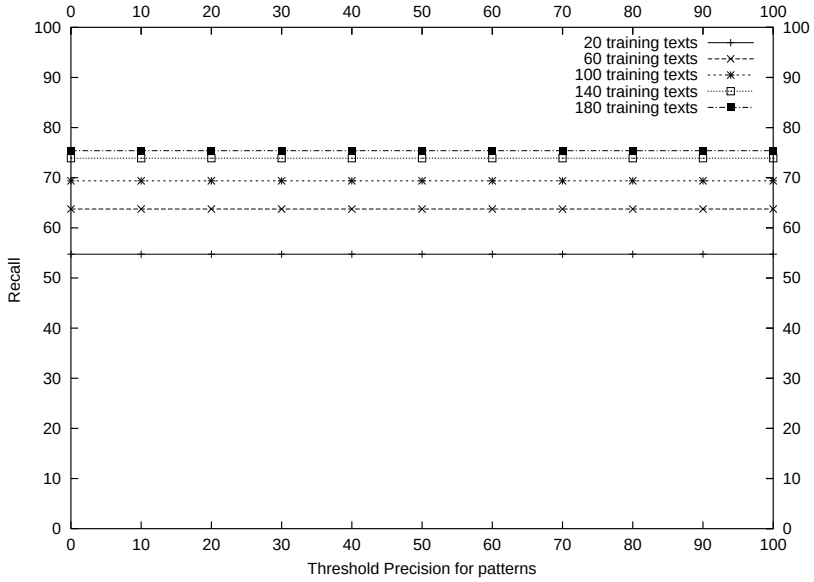


(b)

Figure 6.12: (a) Evolution of recall on test set for patterns learned with estimated precision above a given threshold, for extracting `AIRLINE` information, as the number of training texts is increased, and (b) evolution of recall on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

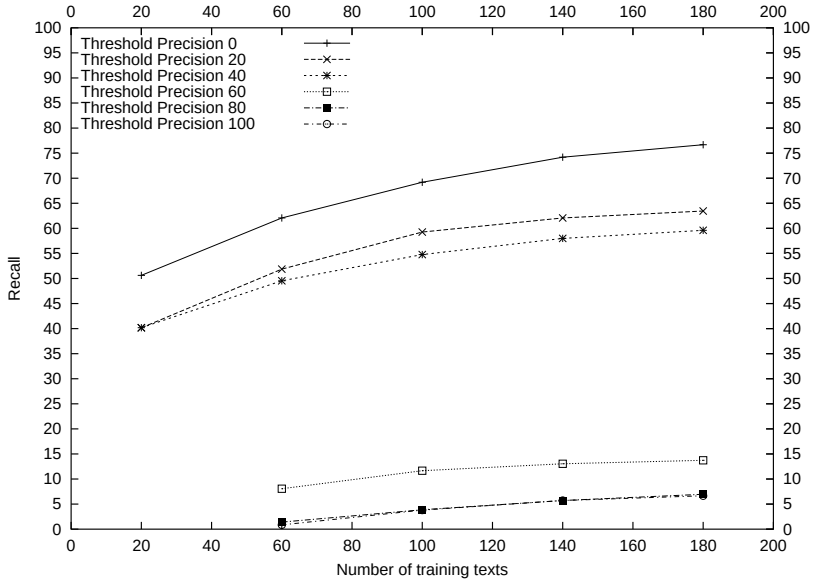


(a)

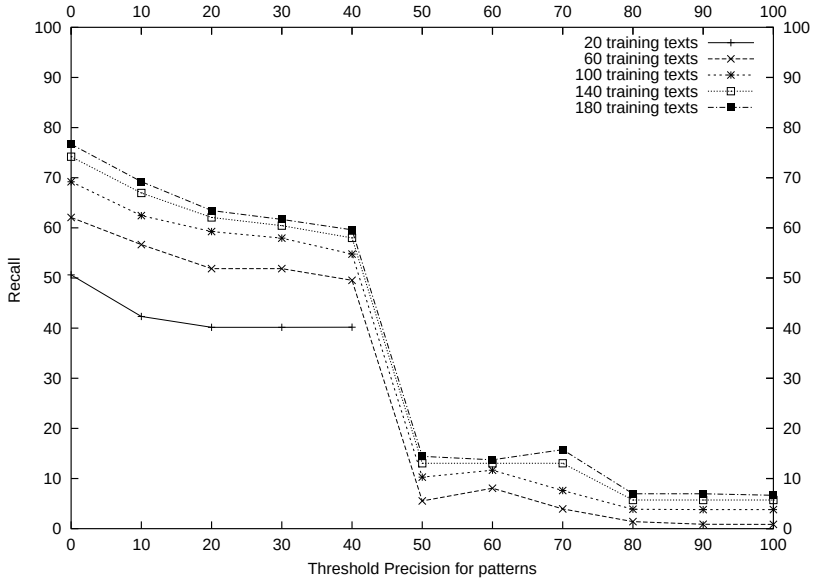


(b)

Figure 6.13: (a) Evolution of recall on test set for patterns learned with estimated precision above a given threshold, for extracting AIRCRAFT information, as the number of training texts is increased, and (b) evolution of recall on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.



(a)



(b)

Figure 6.14: (a) Evolution of recall on test set for patterns learned with estimated precision above a given threshold, for extracting MANUFACTURER information, as the number of training texts is increased, and (b) evolution of recall on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

6.2.3 Evolution of the Precision Measure

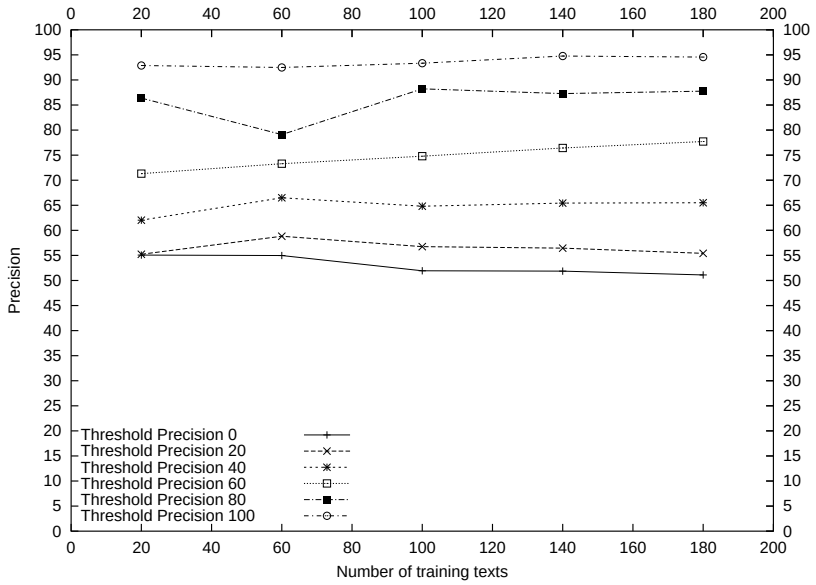
In the same way we did for recall, we now study the evolution of recall of learned patterns as the number of training texts is increased and as the threshold of estimated precision is changed.

Figures from 6.15 (a) to 6.21 (a) show evolution of precision as the training set is increased. Note that, in general, minimum allowed value of estimated precision of 100% does not ensure an actual 100% precision on the test set. It is worth to remember again that when we set a threshold of estimated precision for patterns to be learned, the estimated precision is measured in a sample of the sentences covered by the pattern in the training set. This implies that pattern learned at 100% threshold of estimated precision do not necessarily achieve 100% precision on the test set.

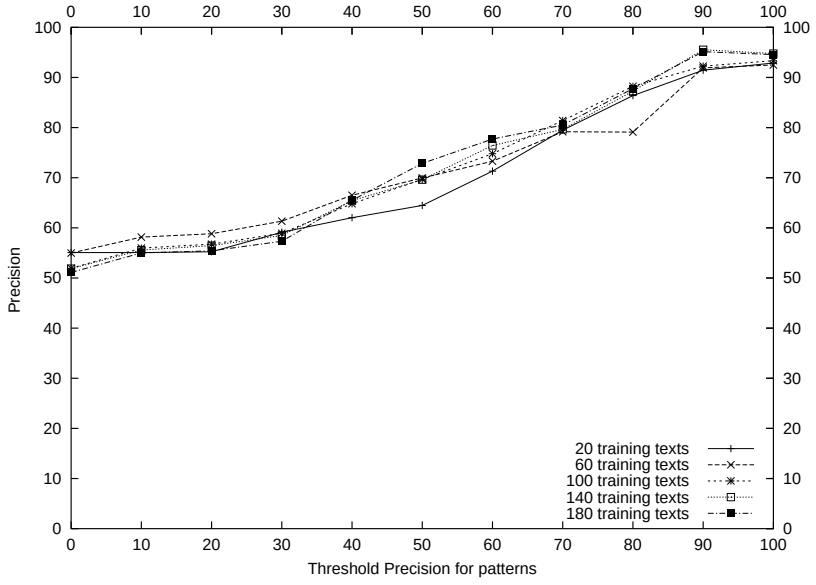
On the other hand, when requiring a minimum estimated precision of 0%, the method usually returns sets of patterns with actual precision on the test set higher than 50%. The reason is that in this set of patterns coexist poorly reliable patterns with very reliable patterns (all of them above 0% of estimated precision).

As it was expected, figures (a) show horizontal lines, which means that threshold of estimation precision is proportional to actual precision on the test set, and that increasing the number of training texts does not have a clear influence in actual precision.

The relationship between precision on test set and minimum estimated precision required on the training set can be also seen in figures 6.15 (b) to 6.21 (b). Figures show how the increase in the threshold of estimated precision increases the precision of the set of learned patterns. They show also the minimum and maximum precision obtained by tuning the threshold parameter.

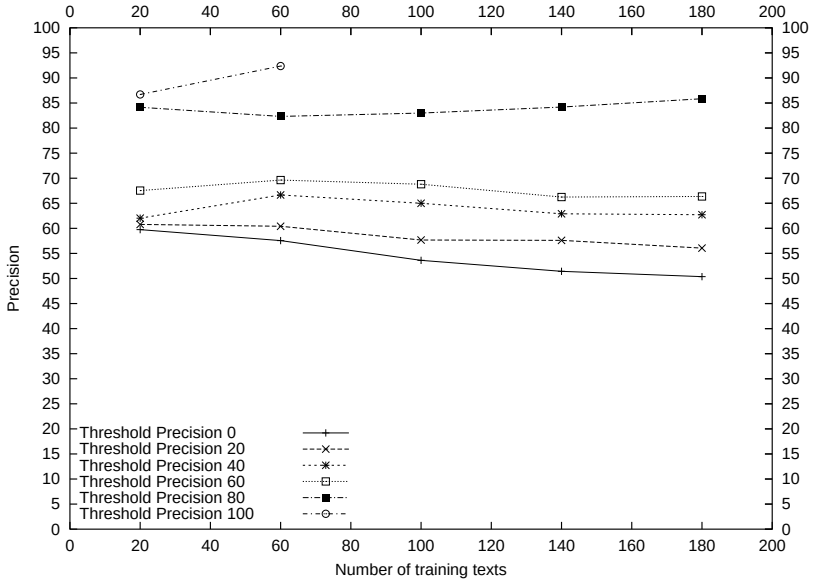


(a)

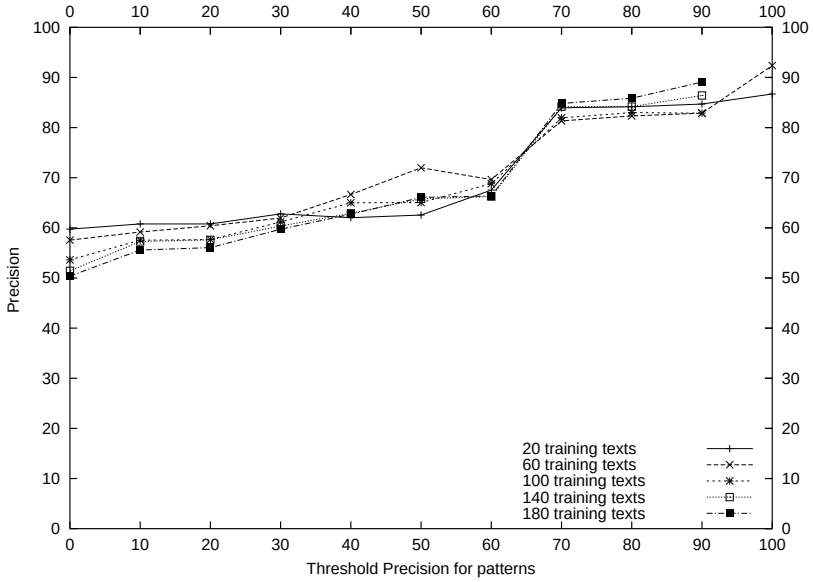


(b)

Figure 6.15: (a) Evolution of precision on test set for patterns learned with estimated precision above a given threshold, for extracting `CRASH-SITE` information, as the number of training texts is increased, and (b) evolution of precision on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

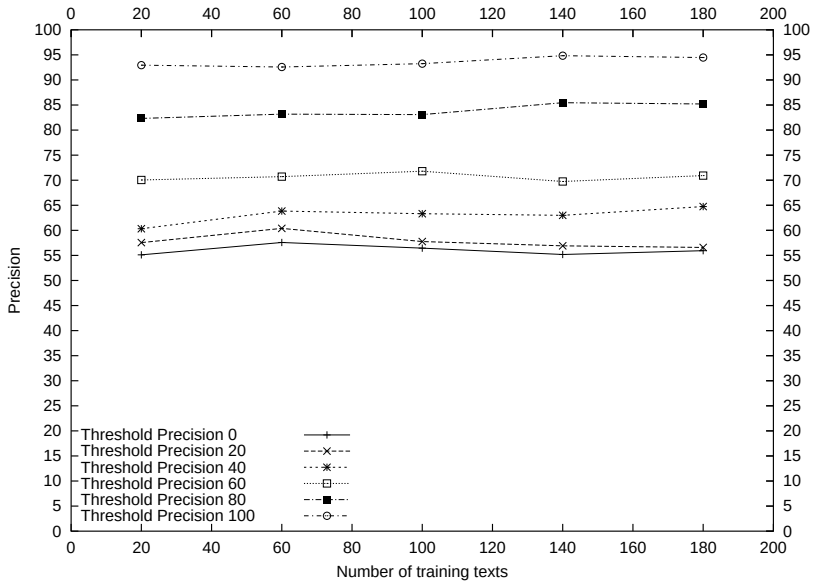


(a)

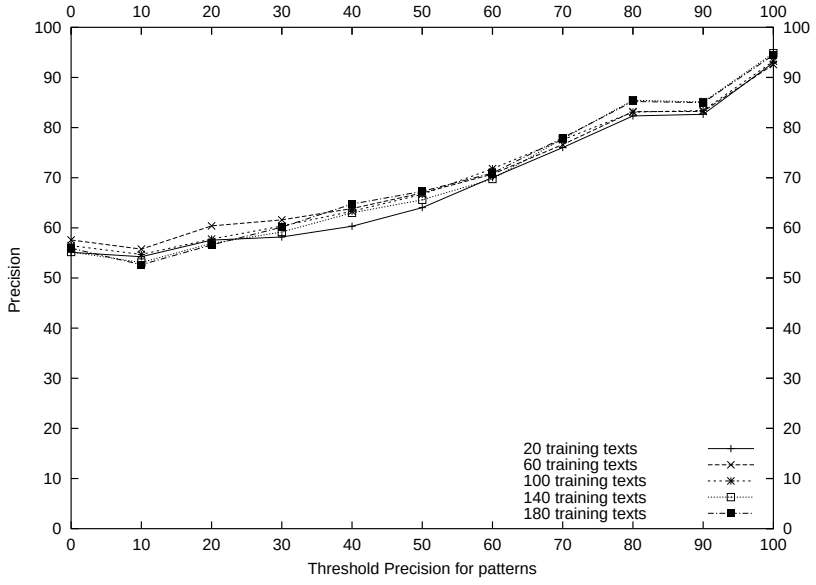


(b)

Figure 6.16: (a) Evolution of precision on test set for patterns learned with estimated precision above a given threshold, for extracting CRASH-DATE information, as the number of training texts is increased, and (b) evolution of precision on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

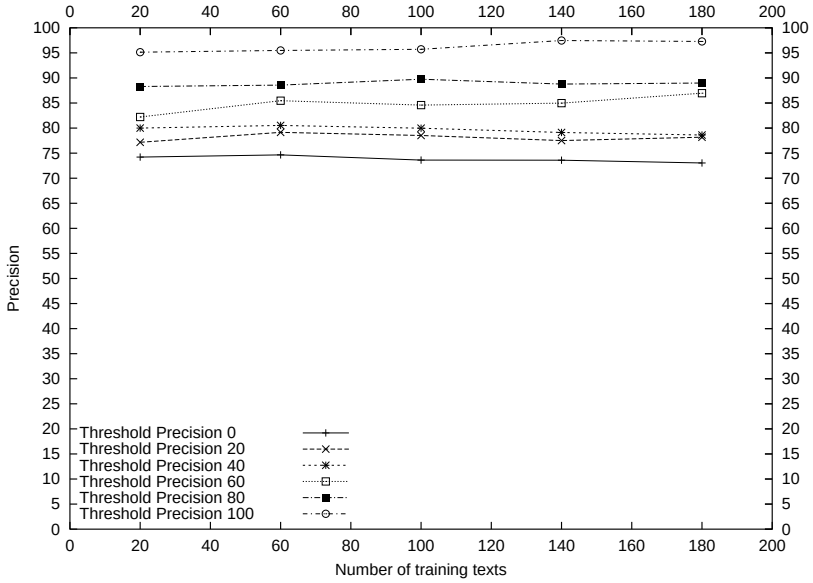


(a)

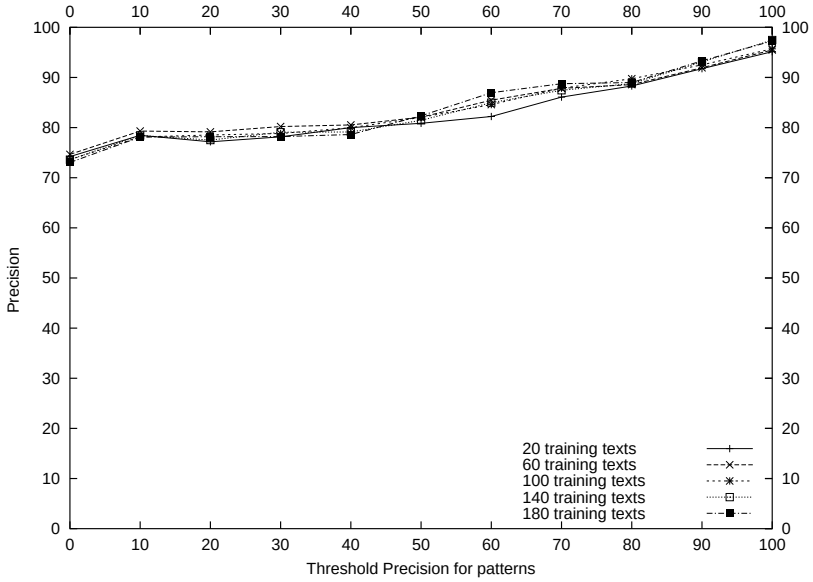


(b)

Figure 6.17: (a) Evolution of precision on test set for patterns learned with estimated precision above a given threshold, for extracting DEPARTURE information, as the number of training texts is increased, and (b) evolution of precision on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

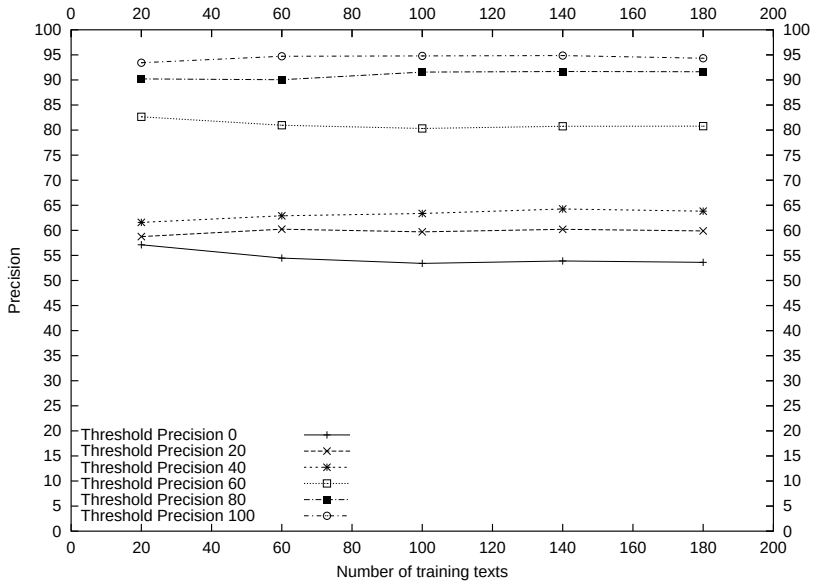


(a)

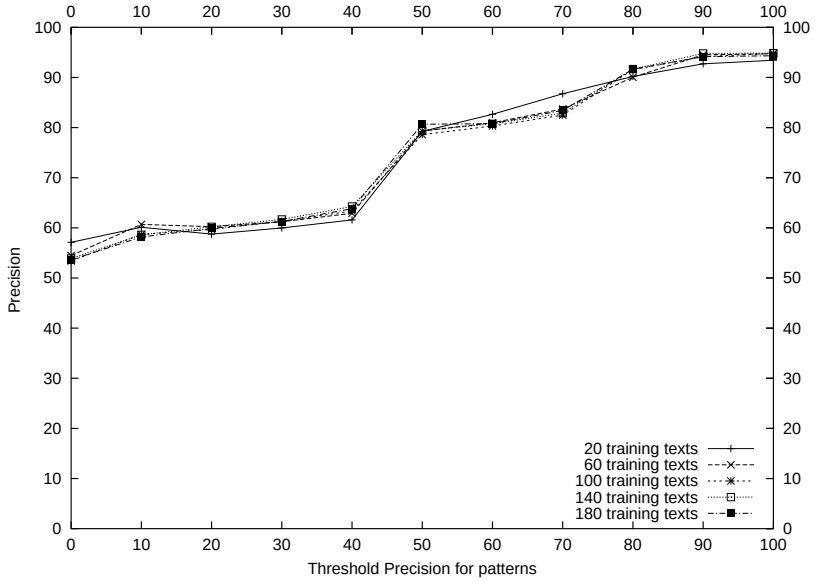


(b)

Figure 6.18: (a) Evolution of precision on test set for patterns learned with estimated precision above a given threshold, for extracting `DESTINATION` information, as the number of training texts is increased, and (b) evolution of precision on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

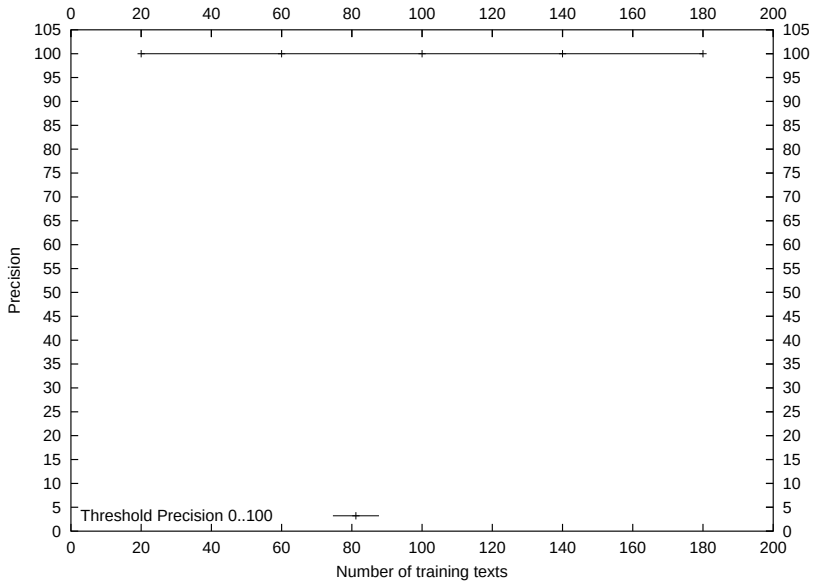


(a)

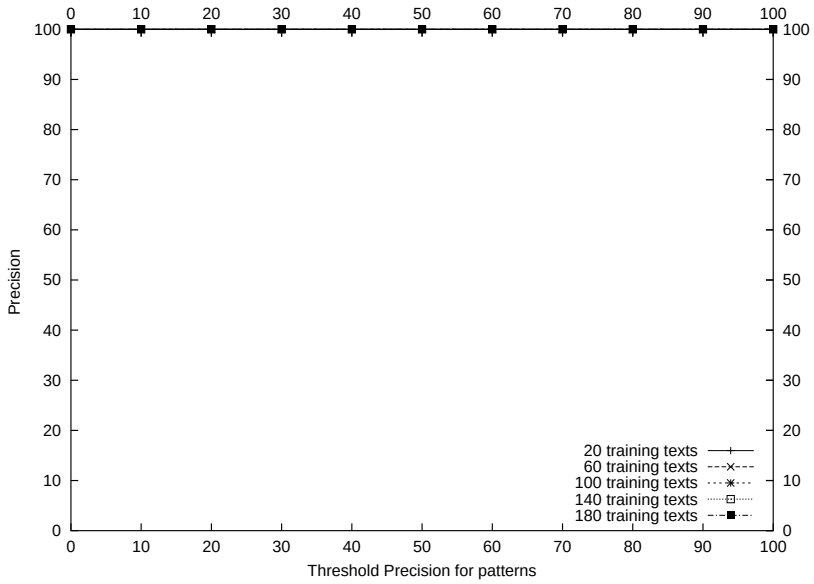


(b)

Figure 6.19: (a) Evolution of precision on test set for patterns learned with estimated precision above a given threshold, for extracting `AIRLINE` information, as the number of training texts is increased, and (b) evolution of precision on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

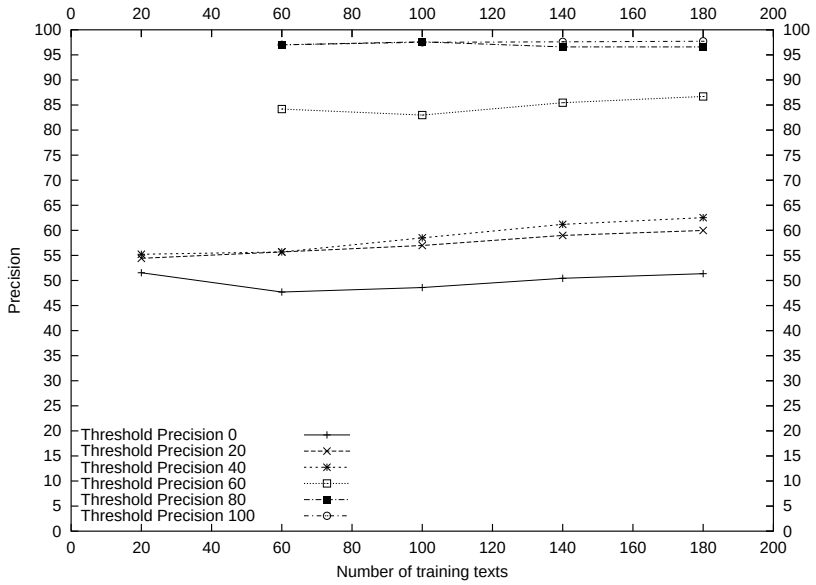


(a)

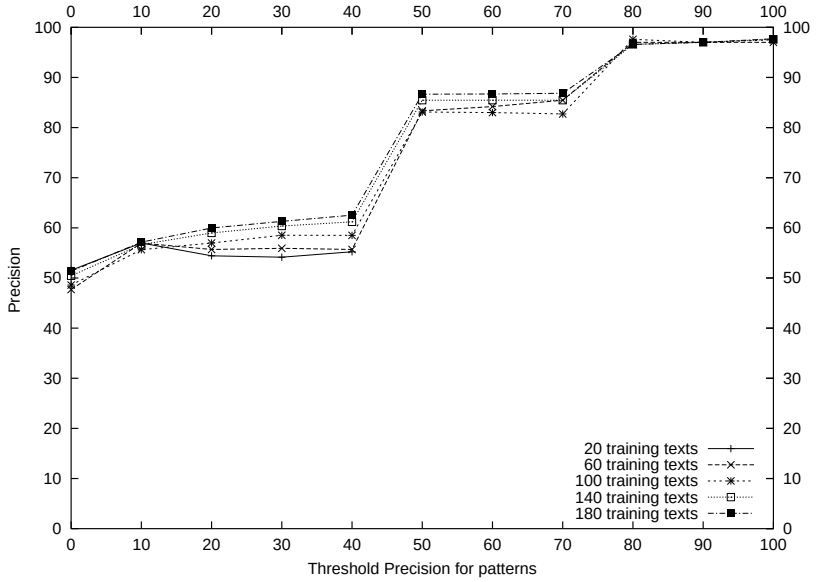


(b)

Figure 6.20: (a) Evolution of precision on test set for patterns learned with estimated precision above a given threshold, for extracting AIRCRAFT information, as the number of training texts is increased, and (b) evolution of precision on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.



(a)



(b)

Figure 6.21: (a) Evolution of precision on test set for patterns learned with estimated precision above a given threshold, for extracting MANUFACTURER information, as the number of training texts is increased, and (b) evolution of precision on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

6.2.4 Evolution of the F measure

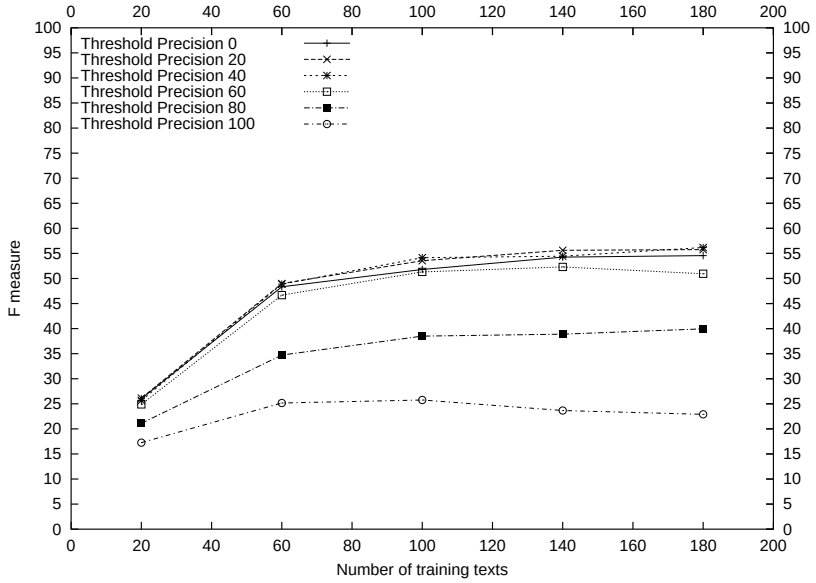
Although it is interesting to study recall and precision scores, the definitive measure that will determine the success of learning is the F measure which balances them (see description in page 19 and formula 2.1). As in the previous cases, figures from 6.22 to 6.28 show the evolution of the F measure as the number of training texts is increased and the threshold of estimated precision is changed.

Figures (a) show clearly that learning curves flattens out by 140 texts for training. After that, more training texts do not significantly improve performance. This indicates that, 140 training texts are enough for this domain.

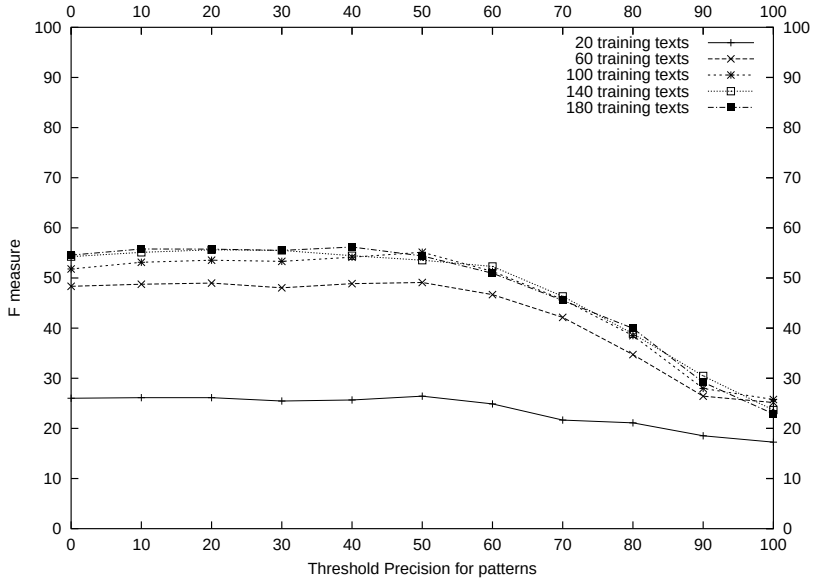
Figures (b) show that, in general, F scores are very similar for thresholds of estimated precision between 0% and 40%, which in addition, are the values achieving better F scores. For higher values of the minimum allowed value of estimated precision on the training set, recall falls too much to be balanced with the increase in precision. Remember that for the set of patterns learned at 0% threshold of estimated precision, precision on the test set varies from 50% to 100%, depending on the slot.

Even more important, these figures also show that threshold of estimated precision is not an unstable parameter. A perturbation of the better value for the threshold implies a very small change in the F score. Thus, it is not critical for the final performance if we choose 40% or 30% for the threshold of estimated precision instead of the best threshold value for the slot.

Finally, figures (b) show that the heuristic procedure that we used in the MUC-like experiments for choosing the best pattern from the list of patterns generated by the *learn-one-pattern* function (see section 5.3), was better than requiring a fixed minimum of estimated precision of patterns. Note that for each slot, figures show that the best F value achieved with 100 training texts (the number of training texts in the MUC-like experiment) was similar but often lower than results reported in the table. The only exception are the results for the `MANUFACTURER` and `AIRLINE` slots, that are not comparable because in the current experiments we are able to also extract information from modifiers.

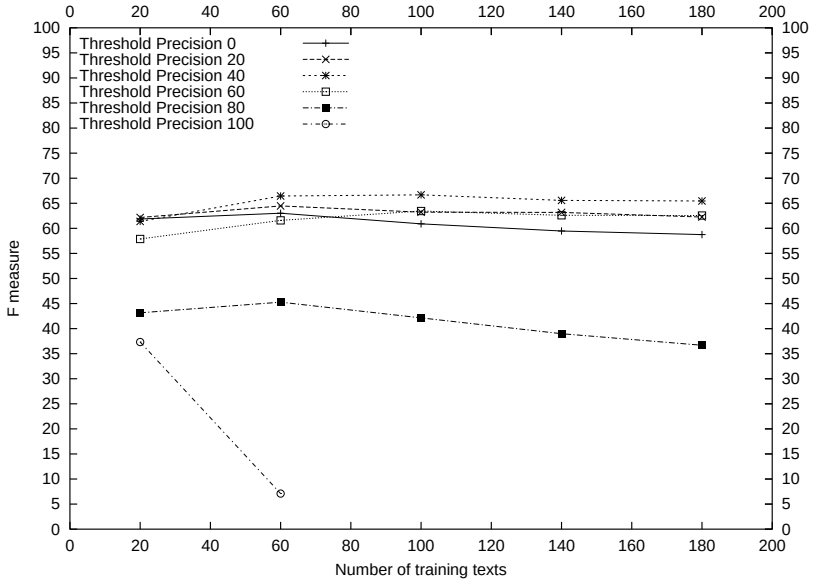


(a)

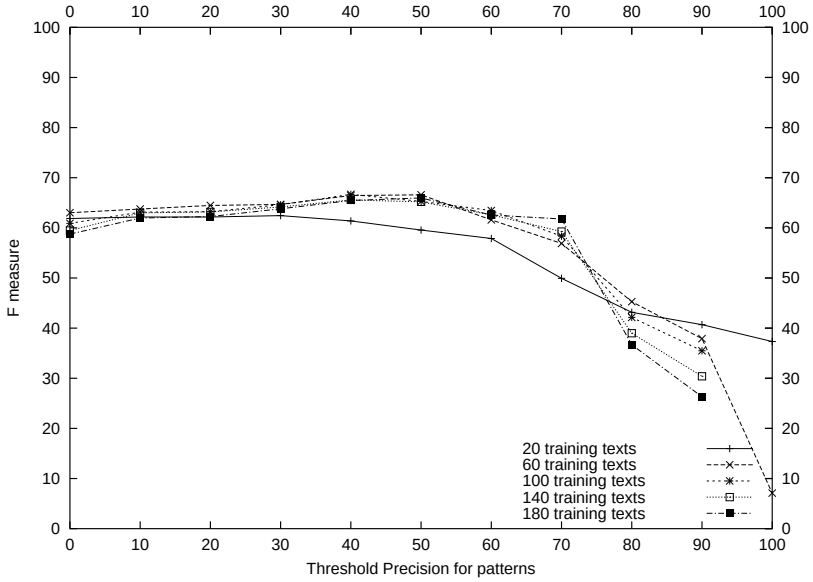


(b)

Figure 6.22: (a) Evolution of F measure with $\beta = 1$ on test set for patterns learned with estimated precision above a given threshold, for extracting CRASH-SITE information, as the number of training texts is increased, and (b) evolution of the same measure on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

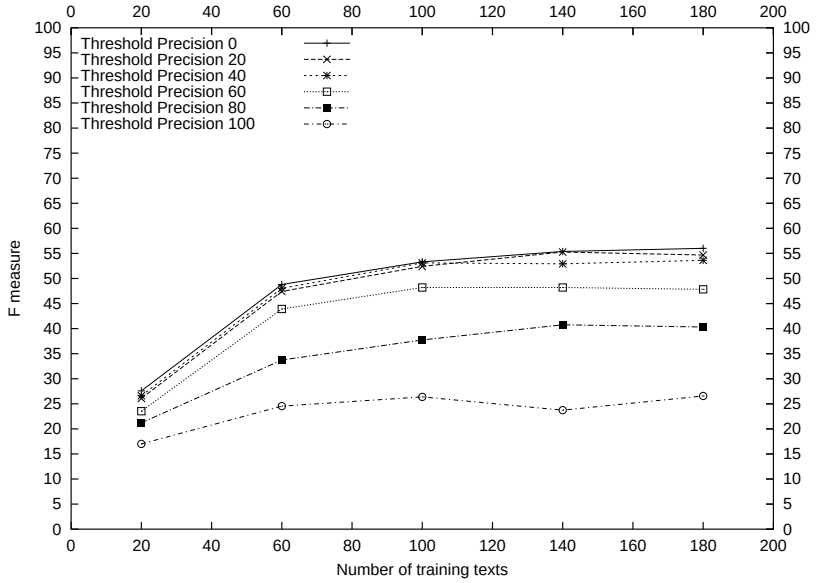


(a)

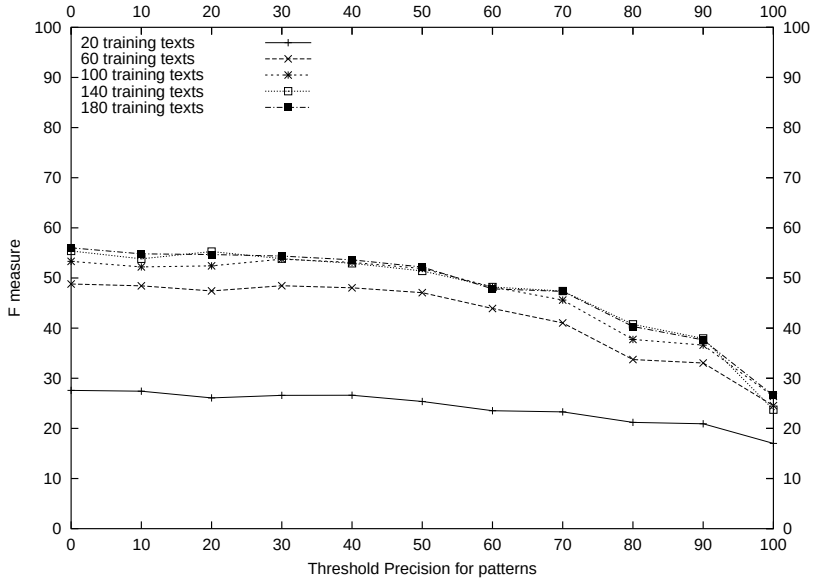


(b)

Figure 6.23: (a) Evolution of F measure with $\beta = 1$ on test set for patterns learned with estimated precision above a given threshold, for extracting CRASH-DATE information, as the number of training texts is increased, and (b) evolution of the same measure on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

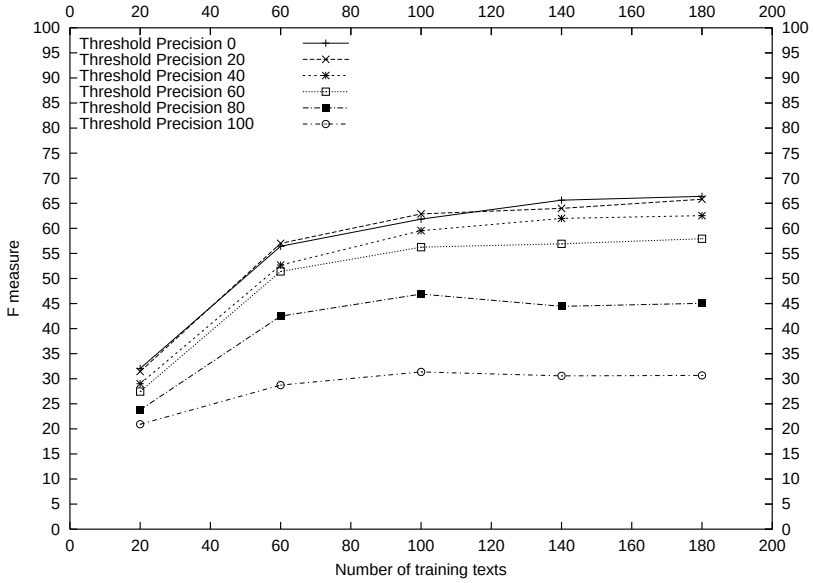


(a)

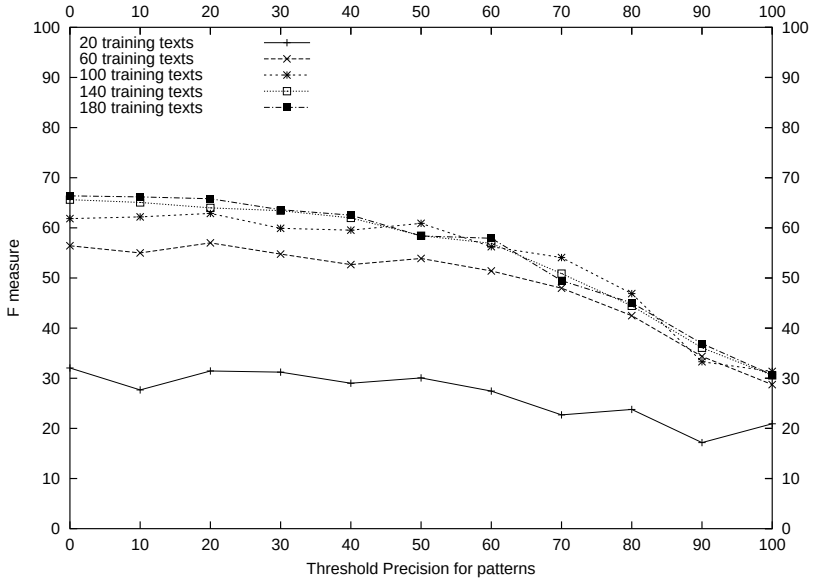


(b)

Figure 6.24: (a) Evolution of F measure with $\beta = 1$ on test set for patterns learned with estimated precision above a given threshold, for extracting DEPARTURE information, as the number of training texts is increased, and (b) evolution of the same measure on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

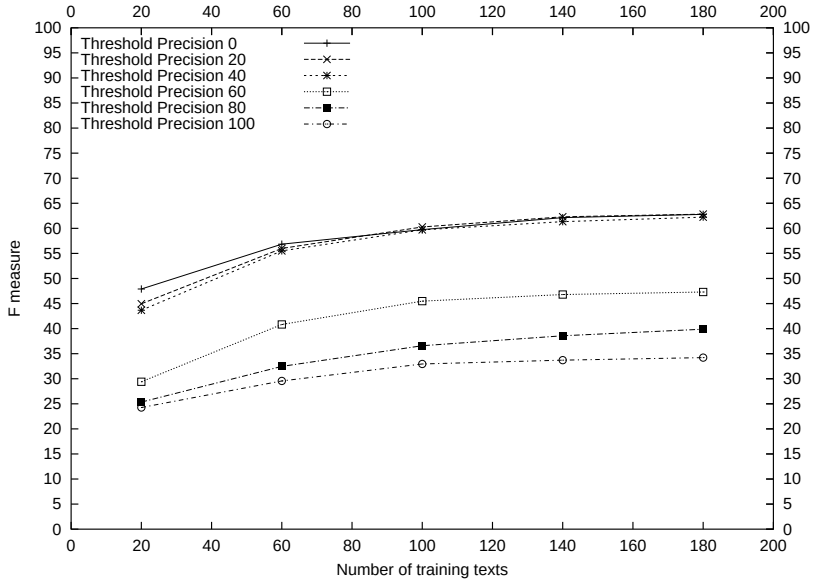


(a)

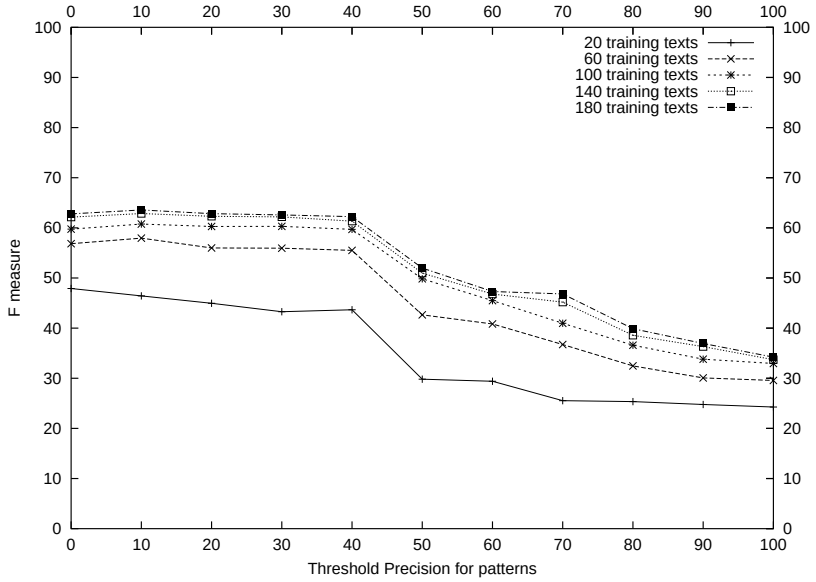


(b)

Figure 6.25: (a) Evolution of F measure with $\beta = 1$ on test set for patterns learned with estimated precision above a given threshold, for extracting `DESTINATION` information, as the number of training texts is increased, and (b) evolution of the same measure on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

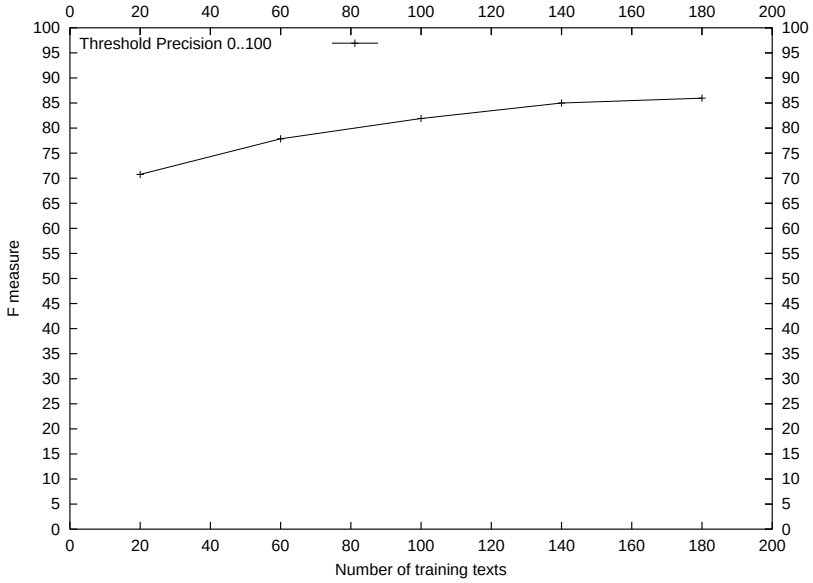


(a)

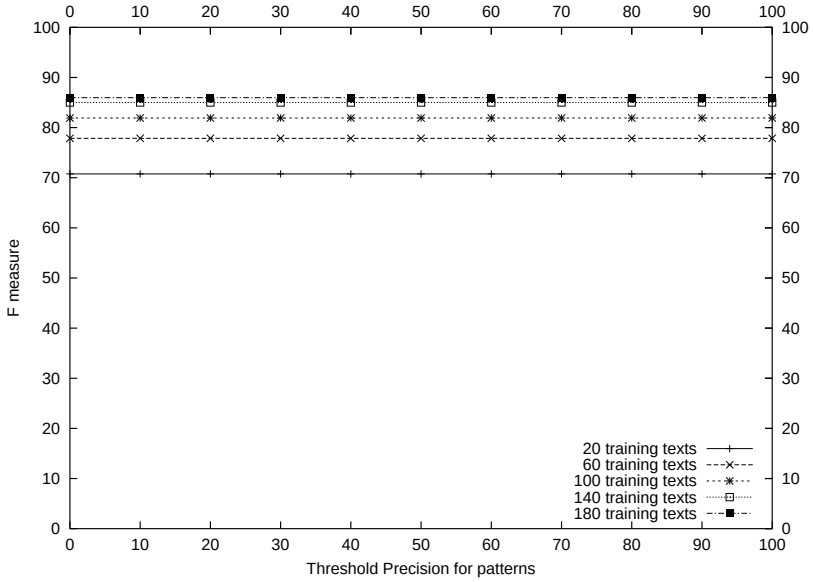


(b)

Figure 6.26: (a) Evolution of F measure with $\beta = 1$ on test set for patterns learned with estimated precision above a given threshold, for extracting AIRLINE information, as the number of training texts is increased, and (b) evolution of the same measure on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

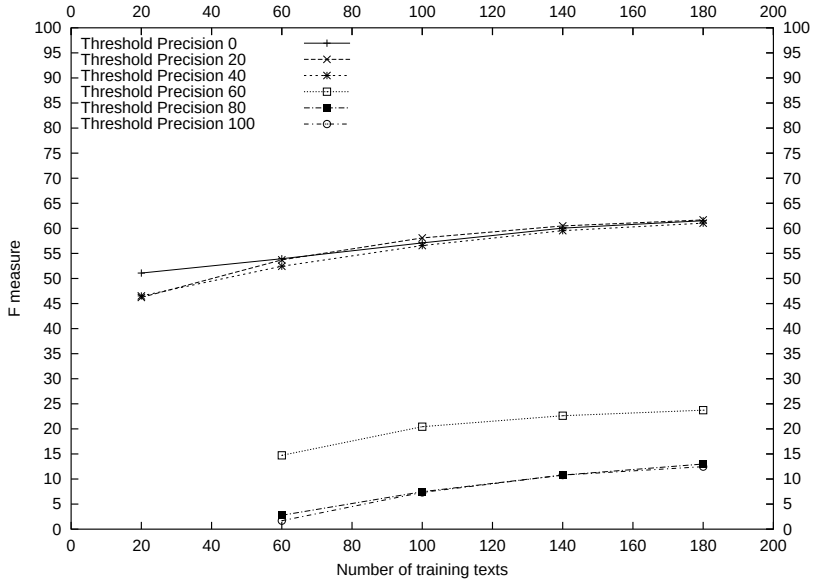


(a)

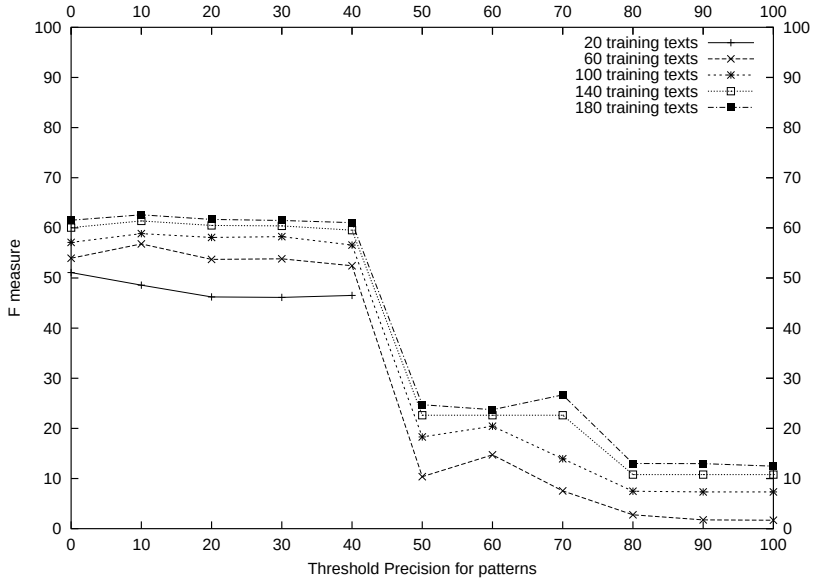


(b)

Figure 6.27: (a) Evolution of F measure with $\beta = 1$ on test set for patterns learned with estimated precision above a given threshold, for extracting AIRCRAFT information, as the number of training texts is increased, and (b) evolution of the same measure on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.



(a)



(b)

Figure 6.28: (a) Evolution of F measure with $\beta = 1$ on test set for patterns learned with estimated precision above a given threshold, for extracting MANUFACTURER information, as the number of training texts is increased, and (b) evolution of the same measure on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased.

6.2.5 Recall-Precision Plots

Other interesting plots that allow us to understand the tradeoff between recall and precision (better than the F measure) are the so called Recall-Precision plots. Figures from 6.29 to 6.35 show this relationship for each slot[2].

The interesting thing of these figures is that they allow us to anticipate the results depending on the user's different needs of recall and precision. Formula 2.1 for the F measure includes a β parameter that weights the measure to prefer better precision or better recall attending to user's interests. With $\beta > 1$ the user cares more about precision than recall. With $\beta < 1$ the user cares more about recall than precision. In all the previous experiments we conducted, we set β to 1, giving in this way the same importance to recall and precision. Note that, in some cases, giving equal importance to recall and precision would not be desirable. For example we could be more interested in extracting accurate information of crash accidents than in extracting all crash information but with a lower precision.

The reported figures can be read in two ways: precision determines recall, and recall determines precision. Let us consider that the user requires 80% precision for the `CRASH-SITE` slot when training on 100 texts. In this case, by inspecting figure 6.29, we see that the system would achieve about 35% recall. On the other hand, if the user were more interested in achieving a high recall for the same slot (close to 60% recall with 180 training texts), the system would obtain close to 50% precision.

Note that these figures do not report values for all the ordinate and coordinate values. For instance, there is no recall info for recall for 20% precision in the plot 6.29 for the `CRASH-SITE` slot. The reason is that, in ESSENCE, whichever value we use to tune the threshold of estimated precision parameter, we never obtain results achieving less than 50% precision for that slot.

One common feature in all plots is that curves describing learning results from 180 training texts are above all the other curves, which means that learning from 180 texts outperforms learning from a smaller amount of training texts. However, curves describing learning results for 140 and 180 training texts are very close, which means that adding 20 new texts to 140 texts for training does not improve performance very much.

[2]Notice that these figures compare precision of patterns on the test set with recall of patterns on the test set, while figures 6.8 to 6.14 showed the relationship between recall on the test set and the minimum allowed value of estimated precision (which is not a measure of true precision).

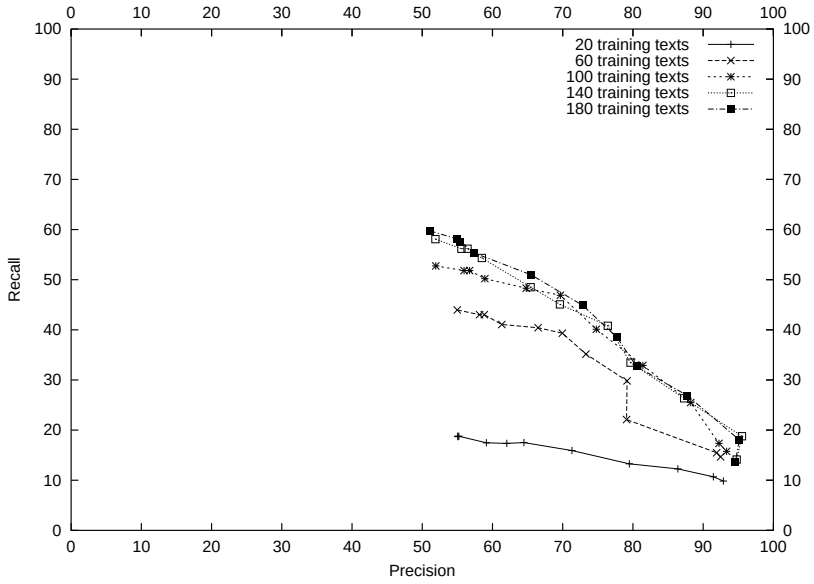


Figure 6.29: Recall-Precision plot for patterns learned from different number of training texts for extracting CRASH-SITE information.

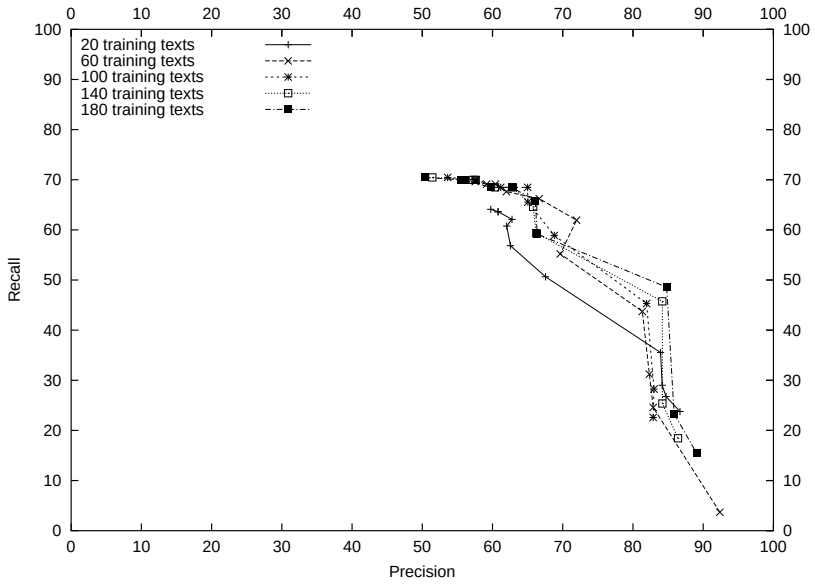


Figure 6.30: Recall-Precision plot for patterns learned from different number of training texts for extracting CRASH-DATE information.

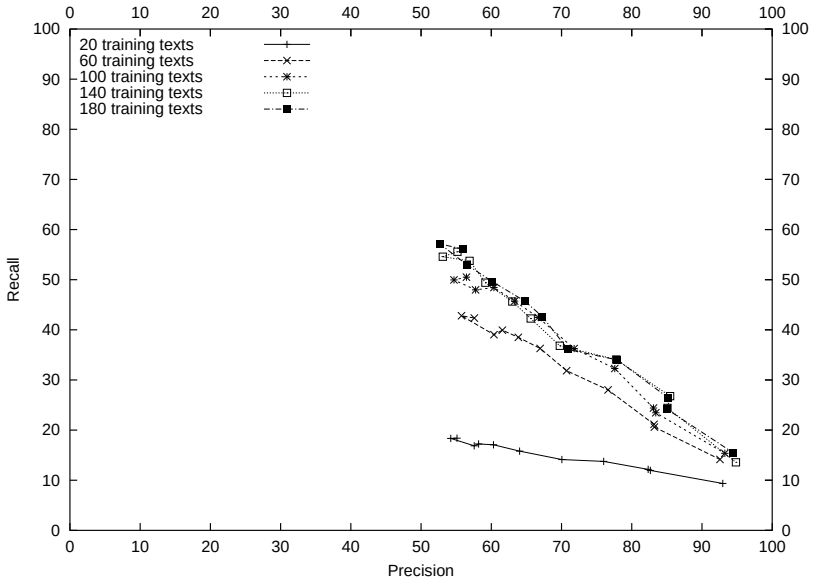


Figure 6.31: Recall-Precision plot for patterns learned from different number of training texts for extracting DEPARTURE information.

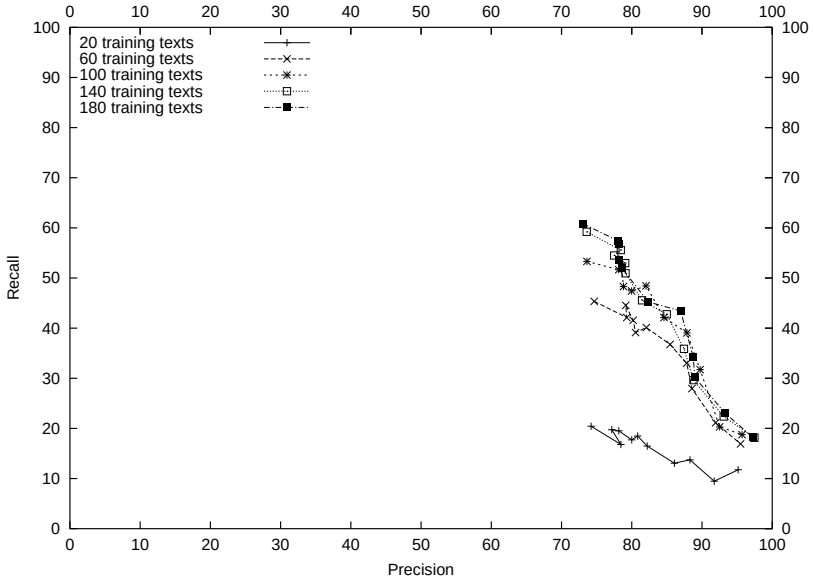


Figure 6.32: Recall-Precision plot for patterns learned from different number of training texts for extracting DESTINATION information.

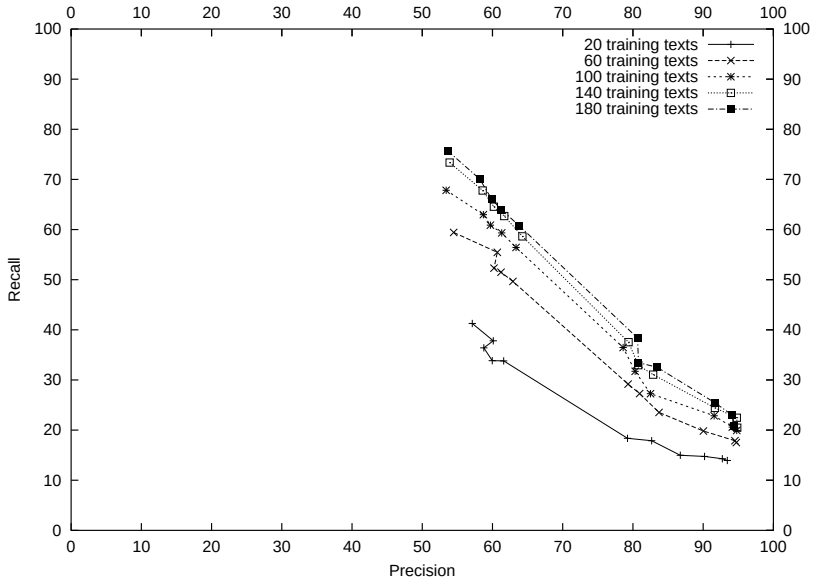


Figure 6.33: Recall-Precision plot for patterns learned from different number of training texts for extracting AIRLINE information.

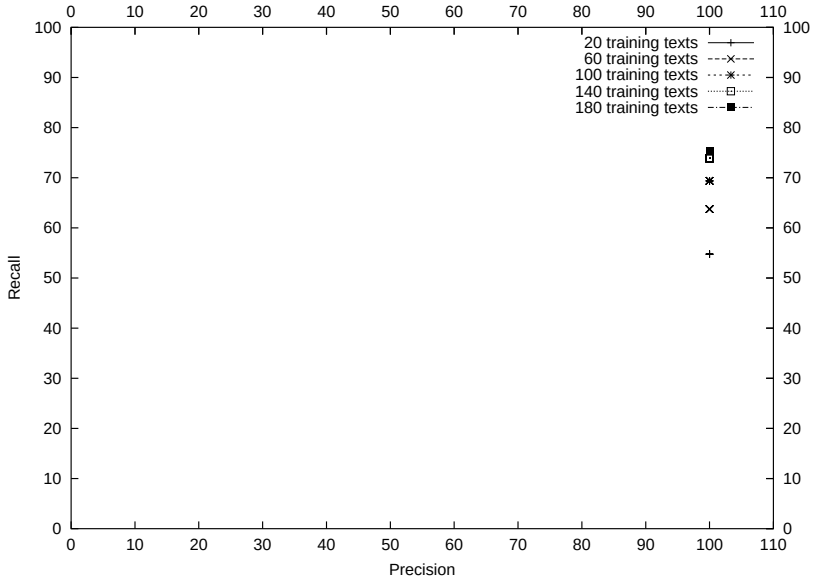


Figure 6.34: Recall-Precision plot for patterns learned from different number of training texts for extracting AIRCRAFT information.

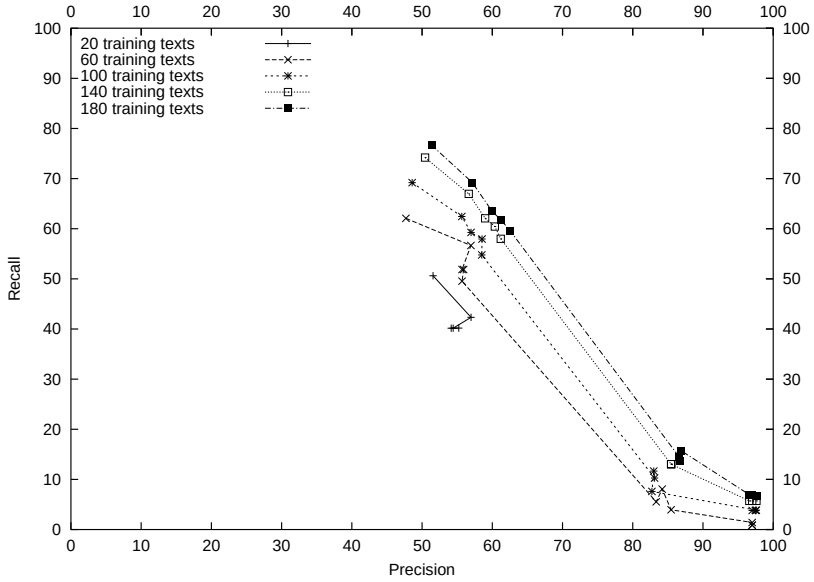


Figure 6.35: Recall-Precision plot for patterns learned from different number of training texts for extracting MANUFACTURER information.

6.3 Effect of Context Window Width

One parameter of the ESSENCE method is the width of the window surrounding the context keyword. An interesting experiment would be to compare results obtained by different values of this parameter. In the previous sections we reported results obtained with window width 5. In this section we report results running ESSENCE for the `CRASH-SITE` slot with window width 7. The evaluation procedure is the proposed in section 6.1.

The expected results should be that increasing the window width we should achieve better recall but worse precision. It remains to be determined to what extent this would happen. Figures 6.36 to 6.40 obtained results. By comparing these figures with those for the same slot shown in the previous sections, we can state that: (1) the overall tendency in plots is the same in both experiments, (2) there has been an increase in recall, (3) the decrease in precision is proportional (but lower) to the increase in recall, and (4) there has been an increase in the F measure, although it is almost imperceptible.

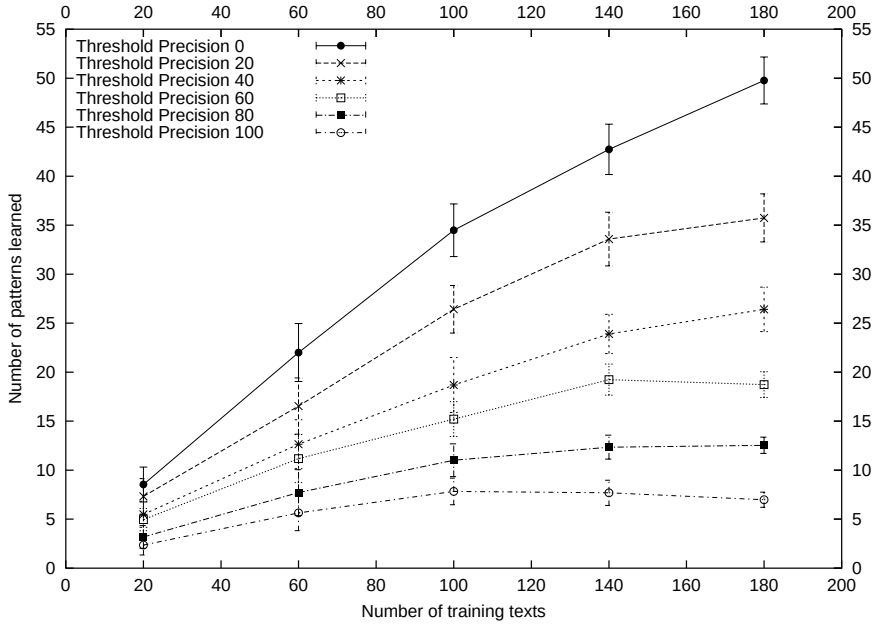
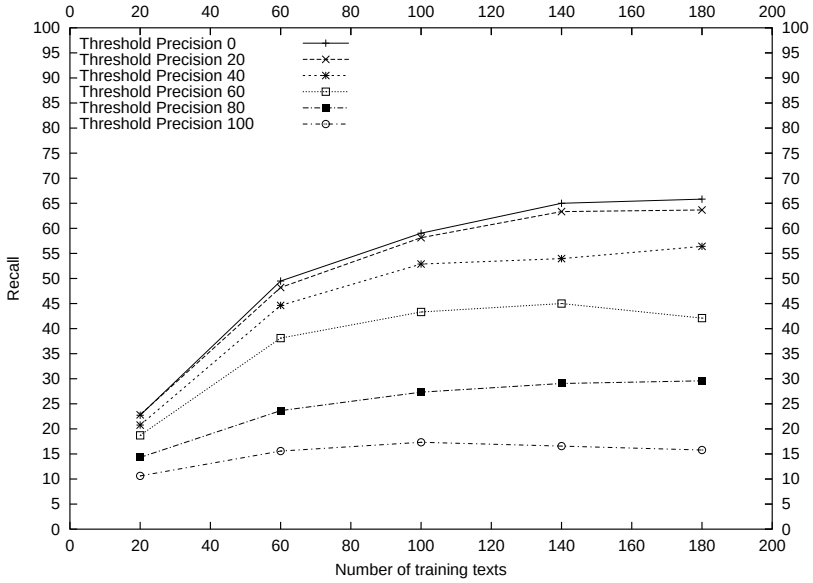
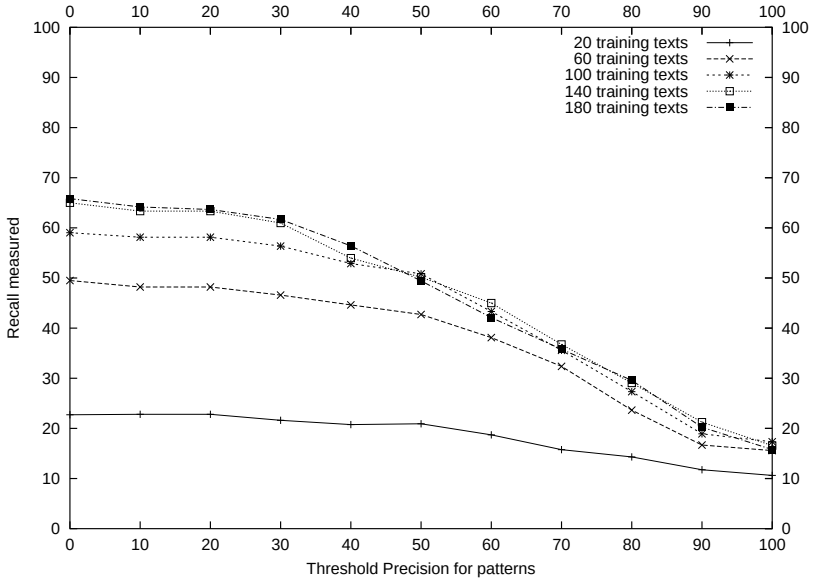


Figure 6.36: Evolution of the number of IE patterns learned above a threshold of estimated precision, for extracting `CRASH-SITE` information, as the number of training texts is increased. Margins show the standard deviation obtained for 20 runs. Window width 7.

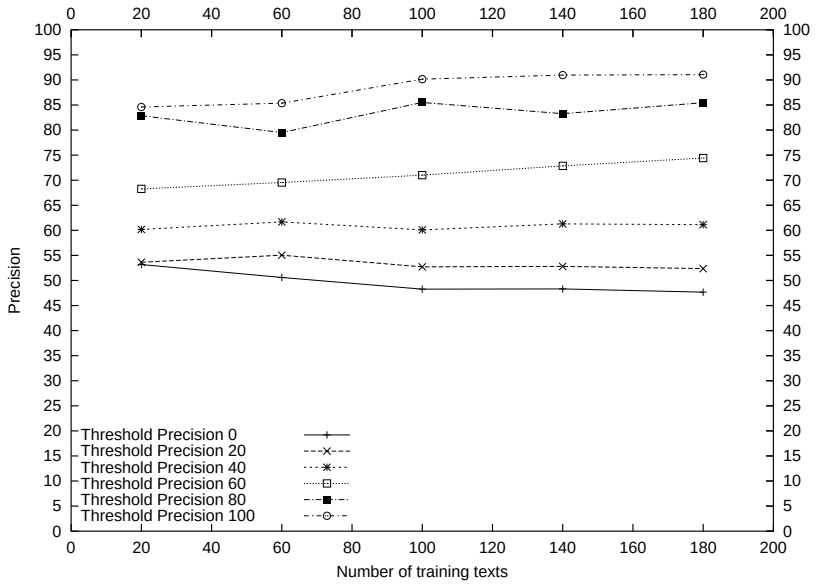


(a)

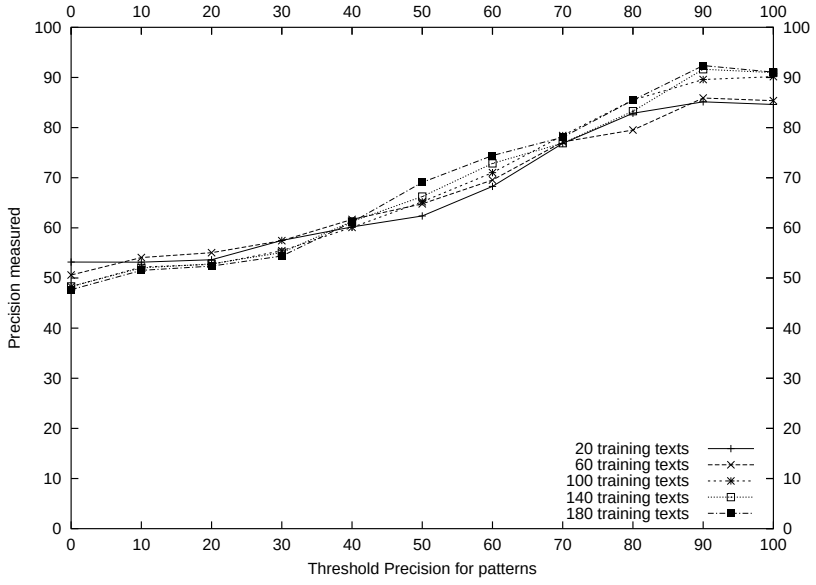


(b)

Figure 6.37: (a) Evolution of recall on test set for patterns learned with estimated precision above a given threshold, for extracting CRASH-SITE information, as the number of training texts is increased, and (b) evolution of recall on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased. Window width 7.

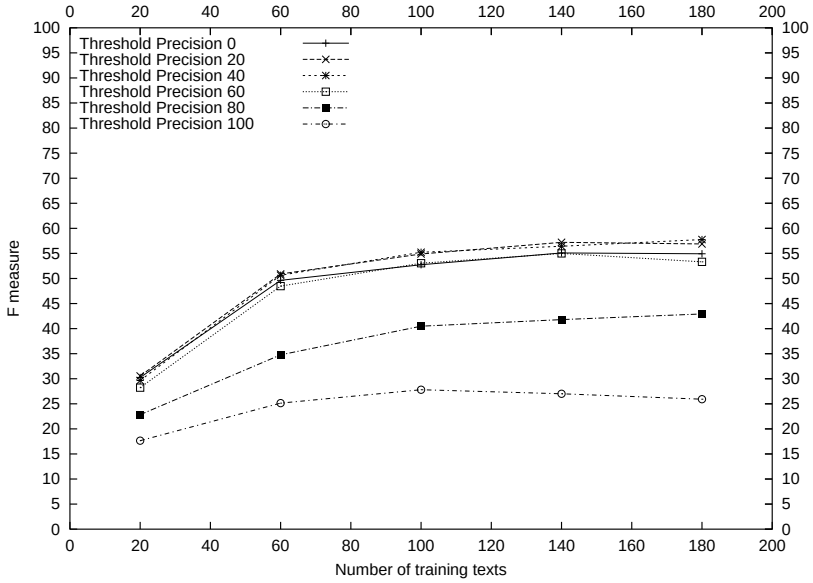


(a)

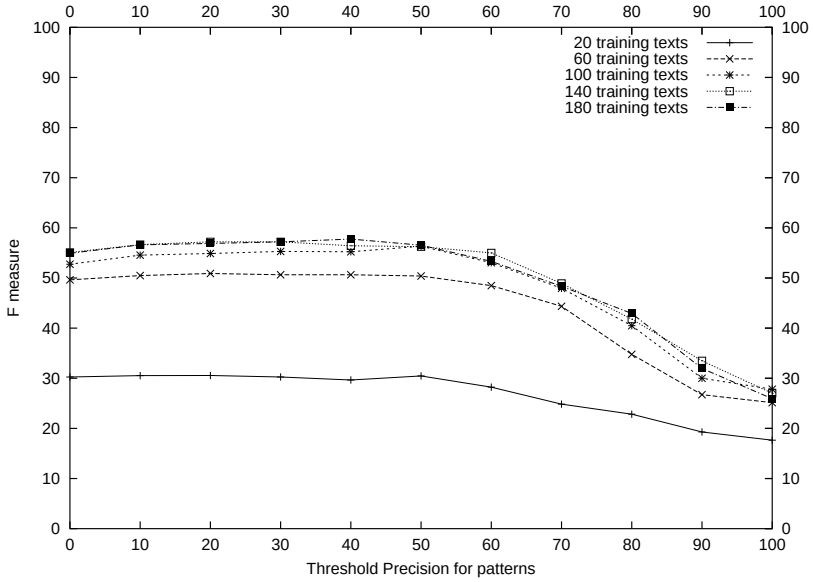


(b)

Figure 6.38: (a) Evolution of precision on test set for patterns learned with estimated precision above a given threshold, for extracting CRASH-SITE information, as the number of training texts is increased, and (b) evolution of precision on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased. Window width 7.



(a)



(b)

Figure 6.39: (a) Evolution of F measure with $\beta = 1$ on test set for patterns learned with estimated precision above a given threshold, for extracting CRASH-SITE information, as the number of training texts is increased, and (b) evolution of the same measure on test set for patterns learned from different number of training texts as the threshold of estimated precision is increased. Window width 7.

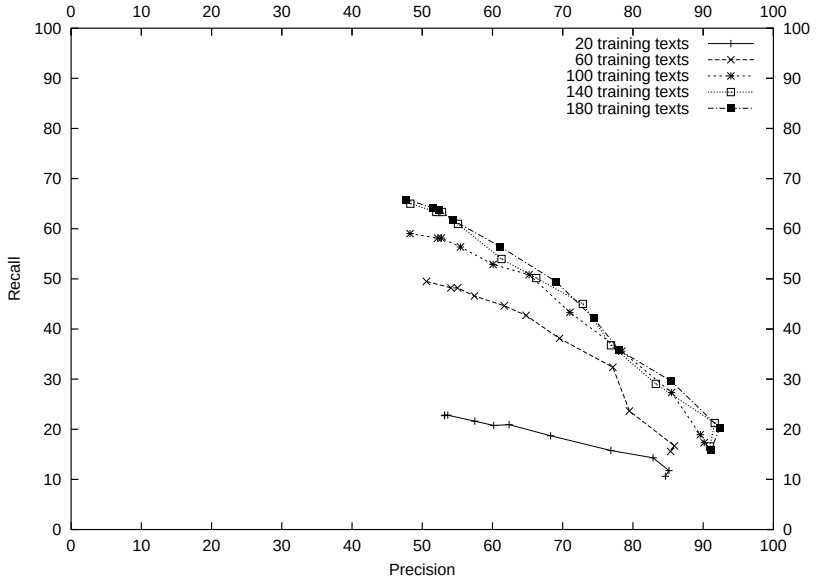


Figure 6.40: Recall-Precision plot for patterns learned from different number of training texts for extracting `CRASH-SITE` information. Window width 7.

6.4 Conclusions

From all the experiments described in this chapter, we conclude the following remarks:

- Results obtained in recall and precision are competitive compared with similar tasks of IE, for example other MUC tasks (MUC 1991; MUC 1992; MUC 1993; MUC 1995; MUC 1998).
- ESSENCE does not require a large training set for learning. A training set containing between 100 and 200 texts is enough for the selected domain.
- The heuristic procedure for selecting the best pattern of the list of patterns generated by the function *learn-one-pattern* has proved effective for learning, and a better procedure than selecting a fixed minimum required estimated precision for all patterns.
- The learning algorithm does not have critical parameters. We have shown that a moderate variation in the best value for the threshold of estimated precision parameter does not affect the final F scores obtained. We also showed that by changing the context window width, the results obtained are very similar.
- Finally, ESSENCE has the ability to deal with recall/precision tradeoff by tuning the threshold of estimated precision parameter.

Chapter 7

Concluding Remarks

This chapter begins with a brief discussion of the feasibility of using WordNet for acquiring IE patterns. This discussion is originated from comments and suggestions provided by anonymous reviewers of this work. This chapter also identifies a number of directions for future research that include improvements to ESSENCE as well as studying the applicability of ESSENCE to other domains and languages.

7.1 Discussion

One issue of our approach that has been discussed in (Grishman et al. 1992) is the adequacy of using WordNet for IE pattern acquisition. IE is a NLP task that is domain specific by definition, while WordNet is a general purpose tool. Since WordNet is a generic resource, it has a limited covering of very specific vocabulary, normally the kind of information to be extracted. Nevertheless, we found WordNet useful for extracting information that involves specific vocabulary. For example, one could wonder whether WordNet could be useful for extracting the `CRASH-SITE` slot[1] since WordNet covering of geographic places is restricted to widely known places. Our experiments on news stories showed WordNet more useful than expected. In the example of the `CRASH-SITE` slot, we found WordNet able to identify most of the places because:

- WordNet had a wider covering than expected, even being able to identify locations in syntactic groups such as “into the Everglades”, “to Colorado Springs”, etc.

[1] See chapter 5 to get information about the `CRASH-SITE` slot and other slots mentioned below.

- Sometimes, locations appear in syntactic groups where the head is a generic place actually covered by WordNet such as “airport” in “to JFK International Airport”, “village” in “in Teton Village”, “mountain” in “into the side of Sheep Mountain”, “inlet” in “off East Moriches Inlet”, “suburb” in “into a Nashville suburb”, etc.

- Usually, in news stories, small locations that are not widely known are accompanied by the name of the state or country in which they are located (and which are covered by WordNet), such as in the following syntactic groups: “in Cali, Colombia”, “to Frankfurt, Germany”, “from Pensacola, Fla.”, etc.

All these examples show how our system is able to extract information about places only using WordNet. Although we could use a gazetteer to solve some cases such as “crashed in Khandahar”, which could not have been extracted by our system because “Khandahar” is not covered by WordNet and because it is not a modifier of a generic head (like mountain, etc), our experience shows that recall will not be improved very much.

7.1.1 Contrast of ESSENCE with Other Related Approaches

In section 2.3 we have reviewed related approaches in automatic IE pattern acquisition. The method we present in this thesis is significantly different from other approaches mainly because it learns without the need of annotated corpus. However, a brief comparison of some features of ESSENCE with similar approaches could help to emphasize other contributions of our approach.

A relevant system related to ESSENCE is PALKA (Kim and Moldovan 1995). In both approaches exists one step (called *frame definition* step in PALKA and called *task definition* step in ESSENCE) in which the expert defines, for each slot to be filled: (1) a semantic category from the ontology and (2) a set of keywords (context keywords in ESSENCE) to focus the search for sentences that might contain information for this slot.

Nevertheless, this step is different in our approach because we use WordNet (not a handmade ontology) for the task definition. This is an important difference because ESSENCE avoids the handcrafting of a special purpose ontology. In addition, ESSENCE allows the automatic extension of the set of keywords in the task definition step, which cannot be done in PALKA. Finally, in the task definition step, ESSENCE allows the definition of multiple semantic tags for the slots of the output template.

Another difference between PALKA and ESSENCE is that PALKA does not allow the generalization of context keywords. This fact increases the num-

ber of patterns to be learned by PALKA when compared with ESSENCE. For example, PALKA requires the patterns <`aircraft` crashed in `location`>, <`aircraft` plunged in `location`> and <`aircraft` slammed in `location`> to express something that can be expressed in a unique pattern in ESSENCE because all these keywords (crashed, plunged and slammed) can be covered by a more general concept.

But the main differences are in the learning algorithm. First, and most important, the ESSENCE learning algorithm learns from an unannotated corpus instead of a set of positive and negative examples. Second, PALKA has specialization procedures to deal with generalizations that include negative examples. These specialization procedures generate disjunctive rules. Since ESSENCE always apply the minimum generalization to build an IE pattern, specialization procedures are not needed and, thus, disjunctive rules are avoided.

Another system that presents some similarities with ESSENCE is *TIMES* (Chai 1998), a further version of the work described by Bagga et el. (1997). This system uses WordNet for generalization as ESSENCE does, but both systems are very different. First, Bagga et al. describes an interface-driven system for building IE patterns: the user has to select seed positive examples from which to build the set of IE patterns, while ESSENCE learns IE patterns without examples. Second, their system does not tackle the problem of disambiguating word senses provided by WordNet (that is solved with a simple heuristic or by the expert), while our generalization procedure automatically disambiguates the sense of a word. Third, their learning procedure does not allow the generalization of context keywords, as in the case of PALKA (see above). Finally, the generalization procedure is totally different. Their approach is top-down, that is, they generalize in the WordNet ontology as much as possible from a single example and then go down in the generalization tree searching for the first pattern with an acceptable precision. ESSENCE implements a bottom-up corpus-driven approach, that is, generalization is performed to cover new sentences in the corpus and from the list of generalizations generated it returns the pattern with maximum relative *F* measure.

7.2 Future Work

This section explains some improvements and extensions that would enhance the performance of the method we propose.

7.2.1 Automatic Detection of Enough Amount of Training Text

In our experiments we have followed the underlying rules of MUC competitions, where usually only 100 texts for training and 100 for testing are given. However, the number of texts available for training and testing would not be so constrained in some other applications. If the number of texts for training were not limited (or very large) it might be possible to measure the influence of new training texts in the generation of IE patterns in order to determine when to stop the learning process. Specifically, if IE patterns are not more generalized after adding several texts, or if new IE patterns are not generated after the addition of several new training texts, we could infer that the number of training texts is sufficient and stop the learning process. This idea could be used to develop an incremental version of the algorithm that prompts the expert to stop feeding the system with new training texts.

7.2.2 Methods for Extending the Set of Context Keywords

Another point that can be improved in our methodology is how to know whether the expert has selected enough context keywords for each slot. This is a critical point common to other approaches that use trigger words. If a useful word for triggering has not been selected as a context keyword, sentences containing it will not be considered for the generation of extraction patterns.

Nevertheless, this problem is reduced in our method because the set of words initially collected by the expert is completed by using synonyms and/or hyponyms/hypernyms from WordNet. This step is not present in other systems that use trigger words, such as PALKA (Kim and Moldovan 1995) and LIEP (Huffman 1995), which extract the set of trigger words from a set of annotated texts but do not extend it further. We think that our approach allows for better recall, because extending the set of context keywords by searching in WordNet hyponyms and hypernyms of known context keywords, extends the set with suitable keywords that do not appear in the training text (but that are synonyms, hyponyms or hypernyms of words that appear in the text). These context keywords cannot be obtained with the usual approaches that only extract keywords from a tagged text.

Finally, if the expert is not sure about the set of context keywords proposed, he/she can browse the corpus in order to evaluate whether the defined set of words is enough. Note that browsing the corpus is much easier and less costly than tagging the corpus.

However, doubt will always remain as to whether there are other undefined words that would be useful for triggering a sentence. One approach to solving

this problem could be searching for words in the corpus that appear near words with meanings that correspond to extracting synsets, that is to say, in reverse: the expert defines extracting synsets, and the system returns words that are usually close to these synsets, defining a suitable measure of closeness. The expert can browse this set of words in order to check whether any of them would be good for triggering.

We are currently studying a bootstrapping approach similar to Riloff and Jones (1999) in order to increment the set of triggering keywords from a set of seed keywords.

The point is that although our method is not able to determine whether there are enough trigger words, there are methods that could be used to automatically extend this set of words.

7.2.3 Porting ESSENCE to new domains and languages

The set of experiments we have conducted aimed to extract information concerning different facts but from a single domain. We chose the dry-run MUC-7 domain due mainly to: (1) the challenge to deal with a "real" domain and IE task, (2) the availability of English free text in a domain and (3) the availability of the answer keys for the scenario template task. We started working on the formal-run MUC-7 domain concerning air vehicle launch reports and updates but our work is still unfinished.

The use of general tools allows ESSENCE to be applied to other languages. At present Spanish WordNet is available (Spanish was included in the EuroWordNet(EWN)[2] project to elaborate a multilingual version of the initial English WordNet). Catalan WordNet developed at the Reference Centre for Linguistic Engineering, financed by the Catalan government (CREL)[3] is also available. These lexical resources, together with the available NLP tools (Màrquez and Padró 1997; Rigau et al. 1997; Cervell et al. 1998; Màrquez and Rodríguez 1998) developed in our research group for Catalan and Spanish[4], will allow the application of the ESSENCE method to IE tasks in these languages.

7.3 Conclusions

Extracting relevant pieces of information is nowadays becoming an important task that can be useful for a large number of applications in many different fields.

[2] http://www.hum.uva.nl/∼ewn

[3] http://www.crel.net

[4] http://www.lsi.upc.es/∼nlp

Electronic text is generating a bulk of data which is difficult to process without the help of systems that ease the access to the information. To build such systems is an arduous task, specially if it is done by hand. For that reason, several approaches to IE system building have begun to use machine learning techniques in various steps of the development process.

This document summarizes our research on the task of acquiring IE patterns for building IE systems. We have presented a new method, called ESSENCE, that performs this task on free text. The main advantage of this method is that it reduces the effort of the expert in the process of developing an IE system, and therefore the cost of production is also reduced. This is achieved by focusing the expert's effort on the definition of the task and on the validation and typification of patterns, while tedious tasks have been automated using of ML techniques, and general purpose linguistic resources and tools.

The linguistic resources are domain independent, for example WordNet and the syntactic analyzer. The independence of the linguistic tools from the IE task facilitates the portability of the method to new tasks and to new domains of extraction.

Moreover, the use of general tools allows the method to be applied to other languages. For example, there is available a multilingual version of WordNet and there are also available NLP tools such as syntactic and morphological analyzers for most of these languages.

7.4 Contributions of the Thesis

This thesis describes ESSENCE, the main contribution of which is a method that tries to reduce the effort of acquiring IE patterns based on ELA, an unsupervised ML algorithm. Nevertheless, different issues have been tackled to attain our purpose and in some of them we have developed new proposals.

ESSENCE shows some novel features which are not present in previous approaches and that constitute the set of contributions of this work. These contributions are summarized next in the same order as they appear in the thesis.

C1: Semantic Definition of the Task

The first step of ESSENCE, involves a definition of the IE task. In this step, the user must define the semantic kind of information that has to be extracted for each slot of the output template. The definition is made by using semantic tags of WordNet (a general-purpose semantic hierarchy) or, when this is not possible, by using specific labels that customized named entity recognizers use to label domain specific information not covered by WordNet.

Although the explicit definition of the task could seem an increase of the work to be done by the expert, it is worth to be done because it produces the following three profitable consequences:

- **Improved precision.** The explicit definition of the kind of information to be extracted allows the system to increase precision scores because it will only learn extraction patterns which extract the right semantic kind of information from the training corpus. For instance, assume we are working on the domain of air crashes as described in newspapers and, in particular, we are interested in extracting information about the crashed aircraft, but that the training corpus also contains texts reporting company crashes in markets. Although descriptions of market crashes use words considered as trigger keywords for aircraft crashes, learning will not take into account sentences describing market crashes because the information attached to the trigger keyword is not of the appropriate kind (aircrafts).

- **Automatic typification.** In unsupervised learning, when one candidate for IE pattern is learned, a human expert must validate it and *specify which specific slots of the output template it can fill*. The last process is named typification of the pattern. When the task is explicitly defined by the expert, this step becomes in most cases automatic, because knowing the semantic kind of information for each slot (provided by the expert in the task definition) and the semantic kind of information the pattern extracts, the link between patterns and slots becomes immediate.

- **Focusing learning only on relevant sentences for the task.** Since the learning process of the method is performed on unannotated text, the system cannot distinguish relevant from irrelevant sentences for the IE task on hand. After task definition, the system can focus learning on relevant sentences for the task, that is, those sentences containing a trigger keyword and having at least one syntactic group carrying the kind of semantic information defined for one slot of the output template. The capability of the system to identify relevant sentences allows the system to learn from a corpus without previously removing non-relevant texts.

C2: Automatic Extension of the Set of Trigger Keywords

ELA, the ESSENCE learning algorithm, bases learning and extraction of information on trigger keywords (which are called *context keywords* in ESSENCE). Other methods in the literature are also based on words for detecting sentences carrying information that must be extracted, for instance (Riloff 1993). Such

approaches has to deal with the problem of getting an adequate set of trigger keys, because when no trigger keys are defined for a sentence carrying relevant information, that information would not be extracted. This fact causes a loss in the recall rate of the information to be extracted.

Usual procedures for defining trigger keys are: asking the expert, collecting words from the expert by browsing the corpus, or learning from the corpus. All these methods can miss some useful trigger words. Specially, note that both browsing and learning methods only collect words that are present in the training corpus.

ESSENCE extends an initial set of trigger keywords defined by the expert[5] by finding synonyms, hypernyms and hyponyms of them in WordNet, and adding the resulting words to the initial set. The enlarged set of keywords is finally filtered by the expert. This procedure allows us to increase recall because the probability of missing a relevant trigger keyword is reduced, and at the same time it allows the automatic definition of new trigger keys not present in the training corpus, for instance synonyms of a valid trigger key. In this way, even from few texts we can find a reasonable set of trigger keywords.

C3: Minimum Handcrafting of Linguistic Resources

As we noted in section 1.1, in order to ease the process of building an IE system and therefore to ease the tuning process needed when porting the system to new user domains, customization of tools and resources should be reduced as much as possible. The main reason for this claim is that these customized tools are handcrafted and therefore expensive.

We propose the use of general purpose tools, like WordNet, that can be used at several steps of the IE system construction process. ESSENCE performs an extensive use of this tool. WordNet gets involved in many processes as: selection of trigger keys, task definition, selection of extracting synsets, parsing, semantic tagging and pattern generalization.

Nevertheless, there may remain some domain vocabulary not covered by such general purpose tools. In these cases, one has to make use of domain specific customized tools.

C4: New Learning Algorithm on Unannotated Text

In order to achieve the main goals of this thesis, described in 1.1, ESSENCE uses a new algorithm for learning IE patterns without user-labeled examples. The learning algorithm, called ELA, focuses on sentences carrying trigger words and finds semantic and syntactic regularities that allow the definition of information extraction patterns.

[5] But also applicable to an initial set of trigger keywords acquired by learning from the corpus.

ELA shows the following distinguishing features in information extraction:

CLA1: Learning without user-labeled examples.

CLA2: It uses semantic knowledge for learning but does not require an explicit Word Sense Disambiguation procedure (neither a human expert for disambiguation).

CLA3: It uses WordNet instead of hand made ontologies.

CLA4: It defines a new measure for closest pattern selection that takes into account the number of matching fields and semantic generalization in fields.

CLA5: Efficient exploration of the corpus by focusing on sentences which contain trigger keywords (or context keywords) and extracting synsets (expected semantic information surrounding a keyword).

ELA is based on a bottom-up construction of candidates for IE patterns from seed sentences. It returns a sorted list of patterns, from the initial seed pattern to the most general pattern. The algorithm shows the following features for selecting the best pattern or rejecting patterns resulted from the learning process:

CLA6: A method to estimate the best pattern in a list of consecutive generalizations.

CLA7: The automatic elimination of patterns that are not enough general.

Bibliography

Eugene Agichtein, Eleazar Eskin, and Luis Gravano. Combining Strategies for Extracting Relations from Text Collections. Technical Report CUCS-006-00. Columbia University Computer Science Department. 2000.

Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plaintext collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*. 2000.

Eneko Agirre and German Rigau. Word Sense Disambiguation using Conceptual Density. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 96)*. 1996.

Jordi Àlvarez. Yet Another Yet Another (YAYA). Technical Report LSI-96-15-T. Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya. 1996.

Jordi Àlvarez. A Description Logic System for Learning in Complex Domains. In *Proceedings of 1998 Workshop on Description Logics (DL'98)*. Trento, June. 1998.

Jordi Àlvarez, Núria Castell, Neus Català, and Àngels Hernández. Control de Calidad de Especificaciones de Software Escritas en Lenguaje Natural. *Actas de I Jornadas de trabajo de Ingeniería del Software* 133–140. 1996.

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, and Mabry Tyson. FASTUS: A Finite-State Processor for Information Extraction from Real-World Text. In *Proceedings of the Thirteen International Joint Conference on Artificial Intelligence*. 1993.

Jonathan H. Aseltine. WAVE: An Incremental Algorithm for Information Extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*. Orlando, Florida, July 19th. 1999.

Jordi Atserias, Núria Castell, Neus Català, Horacio Rodríguez, and Jordi Turmo. Del Texto a la Información. *Novatica* 133:31–35. 1998.

Amit Bagga, Joyce Yue Chai, and Alan W. Biermann. The Role of WordNet in The Creation of a Trainable MESSAGE UNDERSTANDING System. In *Proceedings of the Ninth Annual Conference on Innovative Applications of Artificial Intelligence (AAAI-97)*. 1997.

Ammit Bagga and Alan W. Biermann. Understanding MUC Performance. Technical Report CS-1996-17. Department of Computer Science, Duke University. 1996.

Ammit Bagga and Alan W. Biermann. Analizing the Performance of Message Understanding Systems. Technical Report CS-1997-01. Department of Computer Science, Duke University. 1997.

Roberto Basili, Maria Teresa Pazienza, and Paola Velardi. An empirical symbolic approach to natural language processing. *Artificial Intelligence* 85:59–99. 1996.

Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the Workshop on Computational Learning Theory (COLT 98)*. Morgan Kaufmann Publishers. 1998.

Sergey Brin. Extracting Patterns and Relations from the World Wide Web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, (EDBT'98)*. 1998.

Mary E. Califf. *Relational Learning Techniques for Natural Language Information Extraction*. Doctoral dissertation, Artificial Intelligence Lab. University of Texas at Austin. 1998.

Mary E. Califf and Raymond J. Mooney. Applying ILP-based Techniques to Natural Language Information Extraction: An Experiment in Relational Learning. In *Working Notes of the IJCAI-97 Workshop on Frontiers of ILP*. 1997a.

Mary E. Califf and Raymond J. Mooney. Relational Learning of Pattern-Match Rules for Information Extraction. In *Working Papers of ACL-97 Workshop on Natural Language Learning*, 9–15. 1997b.

Claire Cardie. Empirical Methods in Information Extraction. *AI magazine* 65–79. 1997.

Neus Català. ESSENCE: A Portable Methodology for Building Information Extraction Systems. Technical Report LSI-98-54-R. Department of Computer Science. Technical University of Catalonia. 1998.

Neus Català and Núria Castell. Construcción Automática de Diccionarios de Patrones de Extracción de Información. *Procesamiento del Lenguaje Natural. Actas de SEPLN'97* 21:123–135. 1997.

Neus Català, Núria Castell, and Mario Martin. ESSENCE: a Portable Methodology for Acquiring Information Extraction Patterns. In *Proceedings of the 15th European conference on Artificial Intelligence (ECAI 2000)*. 2000.

Neus Català, Núria Castell, and Mario Martin. A portable method for acquiring information extraction patterns without annotated corpora. *Natural Language Engineering* 9(1):1–29. 2003.

S. Cervell, J. Carmona, L. Màrquez, M.A. Martí, L. Padró, R. Placer, H.Rodríguez, M. Taulé, and J. Turmo. An Environment for Morphosyntactic Processing of Unrestricted Spanish Text. In *Proceedings of ELRA'98*. Granada. 1998.

Joyce Yue Chai. *Learning and Generalization in the Creation of Information Extraction Systems*. Doctoral dissertation, Dept. of Computer Science, Graduate School of Duke University. 1998.

Joyce Yue Chai and Alan W. Biermann. The Use of Lexical Semantics in Information Extraction. In *Proceedings of Workshop on Automatic IE and Building of Lexical Semantic Resources (ACL-97)*, 61–70. Madrid, July. 1997.

Fabio Ciravegna. Learning to tag information extraction from text. In *Proceedings of ECAI-2000 workshop on Machine Learning for Information Extraction*. 2000.

Fabio Ciravegna. $(LP)^2$ an Adaptive Algorithm for Information Extraction from Web-related Texts. In *Proceedings of IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*. 2001.

Fabio Ciravegna, P. Campia, and A. Colognese. Knowledge extraction from texts by SINTESI. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*. 1992.

Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning* 3(4):261–283. 1989.

David A. Cohn, Les Atlas, and Richard E. Ladner. Improving Generalization with Active Learning. *Machine Learning* 15(2):201–221. 1994.

Robin Collier. An historical overview of natural language processing systems that learn. *Artificial Intelligence Review*. 1995.

Robin Collier. Automatic Template Creation for Information Extraction, an Overview. Technical report. Department of Computer Science, University of Sheffield, England. 1996.

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. 1999.

Marco Costantino, Russell J. Collingham, and Richard G. Morgan. Information Extraction in the LOLITA system using templates from financial news articles. In *18th International on Information Technology Interfaces*. Pula. 1996.

Jim Cowie and Wendy Lehnert. Information Extraction. *Communications of the ACM*. 1996.

Mark Craven. Learning to Extract Relations from MEDLINE. In *AAAI-99 Workshop on Machine Learning for Information Extraction*. Orlando. 1999.

Hamish Cunningham. Information Extraction – a User Guide. Technical Report CS-97-02. Institute for Language, Speech and Hearing. Department of Computer Science, University of Sheffield. 1997.

Hamish Cunningham, Robert J. Gaizauskas, and Yorick Wilks. A General Architecture for Text Engineering (GATE) - a new approach to Language Engineering R&D. Technical Report CS-95-21. Institute for Language, Speech and Hearing. Department of Computer Science, University of Sheffield. 1995.

Hamish Cunningham, Yorick Wilks, and Robert J. Gaizauskas. Software Infraestructure for Language Engineering. In *AISB LEDAR Workshop*. Sussex University. 1996.

Luca Dini, Vittorio Di Tomaso, and Frédérique Segond. Error driven Unsupervised Semantic disambiguation. In *ECML'98 Workshop Notes on Towards adaptive NLP-driven systems: linguistic information, learning methods and applications*, 18–29. Chemnitz. Technische Universitat Chemnitz. 1998.

EuroWordNet. Building a multilingual database with wordnets for several european languages. *[http://www.hum.uva.nl/ ewn/] University of Amsterdam*. 1996.

Douglas Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2:139–172. 1987.

Dayne Freitag. Information Extraction From HTML: Application of a General Learning Approach. *Proceedings of the Fifteenth Conference on Artificial Intelligence (AAAI 98)* 517–523. 1998a.

Dayne Freitag. *Machine Learning for Information Extraction in Informal Domains*. Doctoral dissertation, Carnegie Mellon University. 1998b.

Dayne Freitag. Multi-Strategy Learning for Information Extraction. *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 98)* 161–169. 1998c.

Robert Gaizauskas, Takahiro Wakao, Kevin Humphreys, Hamish Cunningham, and Yorick Wilks. University of Sheffield: description of the LaSIE System as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, 207–220. Morgan Kaufmann. 1995.

Robert J. Gaizauskas, Kevin Humphreys, Saliha Azzam, and Yorick Wilks. Concepticons vs. Lexicons: Architecture for Multilingual IE. In *Information Extraction*, ed. Maria Teresa Pazienza. 28–43. Lecture Notes in Artificial Intelligence. Springer-Verlag. 1997.

Oren Glickman and Rosie Jones. Examining Machine Learning for Adaptable End-to-End Information Extraction Systems. In *AAAI-99 Workshop on Machine Learning for Information Extraction*. Orlando. 1999.

Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan Cigarran. Indexing with WordNet synsets can improve text retrieval. In *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, ed. Sanda Harabagiu. Association for Computational Linguistics. 1998.

Ralph Grishman. The NYU system for MUC-6 or where's the syntax? In *Proceedings of Sixth Message Understanding Conference (MUC-6)*, 167–175. Morgan Kaufmann. 1995.

Ralph Grishman. TIPSTER Architecture Design Document Version 2.2. Technical report. Technical report, DARPA (available at http://www.tipster.org/). 1996.

Ralph Grishman. Information Extraction: Techniques and Challenges. In *Information Extraction*, ed. Maria Teresa Pazienza. Lecture Notes in Artificial Intelligence, No. 1714. Springer-Verlag. 1997.

Ralph Grishman, C. Macleod, and J. Sterling. New York university Proteus system: MUC-4 test results and analysis. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, 124–127. 1992.

Ralph Grishman and John Sterling. Generalizing Automatically Generated Selectional Patterns. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*. Kyoto. 1994.

Ralph Grishman and Beth Sundheim. Message Understanding Conference - 6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*. Copenhagen. 1996.

Jerry Hobbs. The Generic Information Extraction System. In *Proceedings of the Third Message Understanding Conference (MUC-5)*, 87–91. Morgan Kaufmann. 1993.

Jerry Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. In *Finite-state Language Processing*, ed. Roche E. and Y. Shabes. 383–406. MIT Press. 1997.

C. Hsu. Initial results on wrapping semistructured web pages with finite-state transducers and contextual rules. In *AAAI-98 Workshop on AI and Information Integration*. 1998.

Scott B. Huffman. Learning Information Extraction Patterns from Examples. In *IJCAI-95 Workshop on New Approaches to Learning for NLP*. 1995.

Paul S. Jacobs and Lisa F. Rau. SCISOR: Extracting Information from On-line news. *Communications of the* ACM. 1990.

Rosie Jones, Andrew McCallum, Kamal Nigam, and Ellen Riloff. Bootstrapping for Text Learning Tasks. In *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*. 1999.

Megumi Kameyama. Information Extraction across Linguistic Boundaries. In *AAAI Spring Symposium on Cross-Language Text and Speech Processing*. 1997.

Vangelis Karkaletsis, Constantine D. Spyropoulos, and Eftihia Benaki. Customising Information Extraction Templates according to Users Interests. In *Proceedings of the International Workshop on Lexically Driven IE*, 23–38. Frascati. 1997.

Leonard Kaufmann and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., NY. 1990.

Jun-Tae Kim and Dan I. Moldovan. Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction. *IEEE Transactions on Knowledge and Data Engineering* 7(5):713–724. 1995.

M. King. Evaluating Natural Language Processing Systems. *Communications of the* ACM. 1996.

George R. Krupka. SRA: Description of the SRA System as Used in MUC-6. In *Proceedings of Sixth Message Understanding Conference (MUC-6)*, 221–235. Morgan Kaufmann. 1995.

Nicholas Kushmerick. *Wrapper induction for information extraction*. Doctoral dissertation, Dept. of Computer Science, U. of Washington, TR UW-CSE-97-11-04. 1997.

Nicholas Kushmerick. Wrapper Induction: Efficiency and Expresiveness. *AAAI-98 Workshop on AI and Information Integration* 82–89. 1998.

W. Lehnert, J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F. Feng, C. Dolan, and S. Goldman. UMass/Hughes: Description of the CIRCUS System used for MUC-5. In *Proceedings of the Fifth Message Understanding Conference*. 1993.

Wendy Lehnert and Beth Sundheim. A Performance Evaluation of Text Analysis Technologies. *AI Magazine* 81–94. 1991.

David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of ICML-94, 11th International Conference on Machine Learning*, ed. William W. Cohen and Haym Hirsh, 148–156. New Brunswick, US. Morgan Kaufmann Publishers, San Francisco, US. 1994.

David D. Lewis and K. Sparck Jones. Natural Language Processing for Information Retrieval. *Communications of the* ACM. 1996.

Xiaobin Li, Stan Szpakowicz, and Stan Matwin. A WordNet-based Algorithm for Word Sense Disambiguation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. 1995.

Alpha Luk, Barbara Vauthey, and Olivier Ansaldi. Acquisition of Domain Specific Salient Semantic Features for Information Extraction. In *Proceedings of the International Workshop on Lexically Driven IE*, 81–90. Frascati. 1997.

Lluís Màrquez and Lluís Padró. A Flexible POS Tagger Using an Automatically Acquired Language Model. In *Proceedings of EACL/ACL-97*, 238–245. Madrid (Spain). 1997.

Lluís Màrquez and Horacio Rodríguez. Automatically Acquiring a Language Model for POS Tagging Using Decision Trees. In *Proceedings of Recent Advances on NLP, RANLP97*, 27–34. Tzigov Chark (Bulgaria). 1997.

Lluís Màrquez and Horacio Rodríguez. Part-of-Speech-Tagging Using Decision Trees. In *Proceedings of 10th European Conference on Machine Learning (ECML 98)*, 25–36. Chemnitz. 1998.

Diana McCarthy. Word Sense Disambiguation for Acquisition of Selectional Preferences. In *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, ed. Piek Vossen, Geert Adriaens, Nicoletta Calzolari, Antonio Sanfilippo, and Yorick Wilks. 52–60. New Brunswick, New Jersey: Association for Computational Linguistics. 1997.

Joseph F. McCarthy and Wendy G. Lehnert. Using Decision Trees for Coreference Resolution. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. 1995.

Ryszard Michalski and R.E. Stepp. Machine Learning: An Artificial Intelligence Approach. In *Learning from Observation: Conceptual Clustering*, ed. J. G. Carbonell R. S. Michalski and T. M. Mitchell. 331–363. Morgan Kauffmann. 1983.

Ryszard S. Michalski. A Theory and Methodology of Inductive Learning. *Artificial Intelligence* 20:111–161. 1983.

Ryszard S. Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. The AQ15 inductive learning system: an overview and experiments. Technical Report UIUCDCS-R-86-1260. University of Illinois. 1986.

Rada Mihalcea and Dan I. Moldovan. Word Sense Disambiguation Based on Semantic Density. In *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, ed. Sanda Harabagiu. 16–22. Somerset, New Jersey: Association for Computational Linguistics. 1998.

George A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*. 39–41. 1995.

Tom Mitchell. Generalization as Search. *Artificial Intelligence* 18:203–226. 1982.

R. Morgan, R. Garigliano, P. Callaghan, S. Poria, M. Smith, A. Urbanowicx, R. Collingham, M. Constantino, and C. Cooper. Description of the LOLITA System as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann. 1995.

Proceedings of the Third Message Understanding Conference (MUC-3). Morgan Kaufmann Publishers. 1991.

Proceedings of the Fourth Message Understanding Conference (MUC-4). Morgan Kaufmann Publishers. 1992.

Proceedings of the Fifth Message Understanding Conference (MUC-5). Morgan Kaufmann Publishers. 1993.

Proceedings of the Sixth Message Understanding Conference (MUC-6). Morgan Kaufmann Publishers. 1995.

On-line Proceedings of the Seventh Message Understanding Conference (MUC-7). http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_proceedings/. 1998.

Ion Muslea. Extraction Patterns for Information Extraction Tasks: A Survey. In *AAAI-99 Workshop on Machine Learning for Information Extraction*. Orlando. 1999.

Ion Muslea, Steve Minton, and Craig A. Knoblock. STALKER: Learning Extraction Rules for Semistructured, Web-based Information Sources. *AAAI-98 Workshop on AI and Information Integration* 74–81. 1998a.

Ion Muslea, Steve Minton, and Craig A. Knoblock. Wrapper Induction for Semistructured, Web-based Information Sources. *Proceedings of the Conference on Autonomous Learning and Discovery (CONALD 98)*. 1998b.

Un Yong Nahm and Raymond J. Mooney. A Mutually Beneficial Integration of Data Mining and Information Extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, 627–632. 2000.

G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An Information Extraction Core System for Real World German Text Processing. In *Proceedings of the Fifth* ANLP. 1997.

Jennifer Norris. Extracting the Essence from Text: a Computational Approach. In *Proceedings of the International Workshop on Lexically Driven IE*. 1997.

Lluís Padró. *A Hybrid Environment for Syntax-Semantic Tagging*. Doctoral dissertation, Department of Computer Science, Technical University of Catalonia. 1997.

Pazienza, Maria Teresa (ed.). *Information Extraction*. Lecture Notes in Artificial Intelligence, No. 1714. Rome: Springer-Verlag. 1997.

Ross Quinlan. Learning Logical Definitions from Relation. *Machine Learning*. 1990.

German Rigau, Jordi Atserias, and Eneko Agirre. Combining Unsupervised Lexical Knowledge Methods for Word Sense Disambiguation. In *Proceedings of joint EACL/ACL-97*. Madrid. 1997.

Ellen Riloff. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI 93)*, 811–816. The AAAI Press/MIT Press. 1993.

Ellen Riloff. Dictionary Requirements for Text Classification: A Comparison of Three Domains. In *AAAI Spring Symposium on Representation and Acquisistion of Lexical Knowledge*. 1995.

Ellen Riloff. Automatically Generating Extraction Patterns from Untagged Text. *Proceedings of the Thirteenth Annual Conference on Artificial Intelligence* 1044–1049. 1996a.

Ellen Riloff. An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. *Artificial Intelligence* 85:101–134. 1996b.

Ellen Riloff. Using Learned Extraction Patterns for Text Classification. In *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, ed. G. Scheler S. Wermter, E. Riloff. 275–289. Springer-Verlag. 1996c.

Ellen Riloff and Rosie Jones. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI 99)*. Orlando. 1999.

Ellen Riloff and Wendy Lehnert. Information Extraction as a Basis for High-Precision Text Classification. *ACM Transactions on Information Systems* 12(3):296–333. 1994.

Ellen Riloff and Jessica Shepherd. A Corpus-Based Approach for Building Semantic Lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, ed. Claire Cardie and Ralph Weischedel. 117–124. Somerset, New Jersey: Association for Computational Linguistics. 1997.

Ellen Riloff and Jay Shoen. Automatically Acquiring Conceptual Patterns Without an Annotated Corpus. In *Proceedings of the Third Workshop on Very Large Corpora*, 148–161. 1995.

RISE. A repository of online information sources used in information extraction tasks. *[http://www.isi.edu/ muslea/RISE/index.html] Information Sciences Institute / USC*. 1998.

Robert E. Schapire. The Strength of Weak Learnability. *Machine Learning* 5:197–227. 1990.

Sam Scott and Stan Matwin. Text Classification Using WordNet Hypernyms. In *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, ed. Sanda Harabagiu. Somerset, New Jersey: Association for Computational Linguistics. 1998.

Satoshi Sekine and Chikashi Nobata. An Information Extraction System and a Customization Tool. In *Proceedings of the New Challenges in Natural Language Processing and its Application*. Tokyo. 1998.

Stephen Soderland. Learning to Extract Text-based Information from the World Wide Web. In *Proceedings of Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*. 1997.

Stephen Soderland. Learning Information Extraction Rules for Semi-structured and Free Text. *Machine Learning* 44(1-3):233–272. 1999.

Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1314–1321. 1995a.

Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. Issues in Inductive Learning of Domain-Specific Text Extraction Rules. In *IJCAI-95 Workshop on New Approaches to Learning for NLP*. 1995b.

Stephen Soderland and Wendy Lehnert. Corpus-Driven Knowledge Acquisition for Discourse Analysis. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. 1994a.

Stephen Soderland and Wendy Lehnert. Wrap-Up: A Trainable Discourse Module for Information Extraction. *Journal of Artificial Intelligence Research* 2:131–158. 1994b.

Stephen G. Soderland. *Learning Text Analysis Rules for Domain-Specific Natural Language Processing*. Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst. 1997.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active Learning for Natural Language Parsing and Information Extraction. In *Proceedings of the Sixteenth International Machine Learning Conference (ICML 99)*, 406–414. 1999.

Jordi Turmo. *An Information Extraction System Portable to New Domains*. Doctoral dissertation, Department of Computer Science, Technical University of Catalonia. 2002.

Jordi Turmo, Neus Català, and Horacio Rodríguez. TURBIO, A System for Extracting Information from Restricted-domain Texts. In *Proceedings of the Eleventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-98-AIE)*. 1998.

Jordi Turmo, Neus Català, and Horacio Rodríguez. An Adaptable IE System to New Domains. *International Journal of Applied Intelligence* 10(2-3):225–246. 1999.

P. Vanderheyden and R. Cohen. Information extraction and the casual user. *AAAI-98 Workshop on AI and Information Integration* 137–142. 1998.

Claudia Wenzel and Michael Malburg. Lexicon-Driven Information Extraction from a Document analysis View. In *Proceedings of the International Workshop on Lexically Driven IE*, 91–102. Frascati. 1997.

Roman Yangarber. *Scenario Customization for Information Extraction*. Doctoral dissertation, New York University. 2001.

Roman Yangarber and Ralph Grishman. Customization of Information Extraction Systems. In *Proceedings of the International Workshop on Lexically Driven IE*, 1–12. Frascati. 1997.

Roman Yangarber and Ralph Grishman. Issues in Corpus-Trained Information Extraction. In *Proceedings of the International Symposium on Spontaneous Speech: Toward the Realization of Spontaneous Speech Engineering*. 2000.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Automatic acquisition of domain knowledge for information extraction. In *In Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*. 2000a.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Unsupervised Discovery of Scenario-Level Patterns for Information Extraction. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP 2000)*, 282–289. Seattle. 2000b.

David Yarowsky. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL 95)*, 189–196. 1995.

Klaus Zechner. A Literature Survey on Information Extraction and Text Summarization. Term paper, Carnegie Mellon University. Available at http://www-2.cs.cmu.edu/ zechner/publications.html, 1997.