Académie de Montpellier

Université **M**ontpellier **2**

Sciences et **T**echniques du **L**anguedoc

Ph.D Thesis

To obtain the
Ph.D. degree of the University Montpellier 2

Discipline	:	**Computer Science**
Doctoral Speciality	:	**Computer Science**
École Doctorale	:	**Information, Structures, Systèmes**

Locating Information in Heterogeneous Log Files

presented and publicly defended by

Hassan Saneifar

December 2, 2011
in front of the jury composed of

To my beloved parents
Mohammad-Taghi and Effat

Contents

Acknowledgement

A thesis does not only involve a scientific research work, but it is also a human experience. During these years, I was lucky to know and work with people that I admire.

I owe my greatest gratitude to my Ph.D. supervisors, Prof. Pascal Poncelet and Dr. Mathieu Roche. I thank them for their support, their enthusiasm, their accurate advices, and the precious knowledge that they transmitted to me. Moreover, they were always present during these years even in situations which were not directly related to my thesis. I admire their professionalism as well as their human side. It was a pleasure for me to have them as advisor. Thank you Pascal and Mathieu.

I would like also to express my gratitude to my industrial supervisor, M. Stéphane Bonniol for his support, his advices and for helping me to know the constraints and challenges in industrial domains. Doing my thesis on an industrial issue gave me the interest for the applied research. Also, I thank him for helping me to resolve many administrative matters. Thank you Stéphane.

It is also my great honour to have Prof. Fabio Crestani, Prof. Eric Gaussier, Prof. Patrick Gallinari, and Prof Violaine Prince as my thesis committee members. I would like to thank them for the valuable time that they have devoted to evaluate this work.

My sincere thanks also goes to Prof. Maguelonne Theisseir, Prof. Anne Laurent, and Dr. Sandra Bringay for their encouragements and specially for having followed my work during my Master. It was delighted to be a member of TaToo project-team. I would also like to thank M. Michel Tabusse for giving me the possibility to carry out this work in Satin Technologies. I always appreciated to work there.

I am indebted to many of my friends and colleagues to support me. I would like to thank especially Dr. Yoann Pitarch, Dr. Julien Rabatel, Dr. Paola Salle, Dr. Lisa Di-jorio, Dr. Cecile Low-Kam, (yes I know, you have all became doctor before me, and it is not fair !), Yuan Lin, Guillaume Artignan, Ghulam Mahdi, Madalina Croitoru. I shared the memorable moments with you all in and out of the LIRMM laboratory. I also thank my colleagues at Satin Technologies: Dominic Spring, Benjamin Montes, Romain FAVAND, Aye-Deen Damba, Amine Douaissia. I appreciated working with you all. I also thank all my other friends and colleagues at LIRMM with whom I shared the different kinds of discussion as well as fun moments.

During my thesis, out of the work, I had few close friends who made me the delicate and difficult moments fun. I am not going to name them here, as they know that I am talking about them and surely neither me nor they will never forget these years. Thank you buddies.

Lastly, and most importantly I wish to thank my family with all my heart. I cannot find the right words to thank my dear brother, Mahdi Saneifar, and my beloved sister, Roghayeh Saneifar, for giving me the chance to have two true friends in life. You guys, you have always supported, encouraged, and loved me even I was most of the time unavailable for you. I feel lucky to have you as brother and sister.
I am deeply indebted to my parents, Mohammadtaghi Saneifar and Effat Sanei. They bore me, raised me, supported me, taught me, and loved me. I would definitely not where I am today without my parents. I sincerely thank them for their love, their unconditional and constant support, for giving me the courage and hope in difficult moments, and for being always present. Hope you will find my deepest gratitude in this words. I love you.

"*Central to this type of thinking is the underlying notion of 'truth'. By means of argument which maneuvers matter into a contradictory position, something can be shown to be false. Even if something is not completely false, the garbage has to be chipped away by the skilled exercise of critical thinking in order to lay bare the contained truth.*"
Edward De Bono

Knowing, it is not disassembling or explaining. That is access to the vision.
Antoine de Saint-Exupéry

Chapter

1 Introduction

If we knew what it was we were doing, it would not be called research, would it?
Albert Einstein

Preamble

This chapter is devoted to introduce the context, motivations, objectives, and the data type that we deal with, i.e., log files. We first present the general notion of log files. Then we introduce the context of this work. A presentation of research domains involved in this work, notably Question Answering Systems, is subsequently provided. We finally present a particular type of log files that we deal with in this research project. We study in detail the specificities of these log files.

Contents

1.1 Introduction

Nowadays, in many application areas, modern computing systems are instrumented to generate huge reports about occurring events in a format of textual data usually called log files. Log files are generated in every computing field to report the status of systems, products, or even causes of problems that can occur.

Number of computational systems which output hundreds of log files, documenting what they are doing and how the tasks are performed, is dramatically increasing. In many application areas like as digital design or monitoring systems, it is not unusual that gigabytes of log files to be generated per day.

Log files may also include data about critical parameters, sensor outputs, or a combination of those. Such files are also used during various stages of software development, mainly for debugging and profiling purposes. Log files became a standard part of large application and are essential in operating systems, computer networks, and distributed systems.

Analysing log files, as an attractive approach for automatic system management and monitoring, has been enjoying a growing amount of attention [Li et al., 2005]. Although the process of generating log files is quite simple and straightforward, log file analysis could be a *tremendous* task that requires *enormous computational resources*, *long time* and *sophisticated procedures* [Valdman, 2001-04]. Indeed, there are many kinds of log files generated in some application domains which are not systematically exploited in an efficient way because of their special characteristics.

There are also many proposals to standardize log formats such as W3C and SNMP formats [Jiang et al., 2008]. To generate Web server log files, according to general use of Web servers, there is a universal format. The standardization efforts concerning log files also resulted in (today expired) IETF draft that proposes the Universal Format for Logger Messages, ULM. The ULM format is a set of guidelines to improve semantic of log messages without exact formalization [Valdman, 2001-04]. However, most log files generated in other fields use *ad-hoc non-standardized* logging formats.

1.2 Context

There are different types of log files based on the application domain. In this thesis, we are mainly interested in log files generated by Electronic Design Automation (EDA) systems. Electronic design automation is a category of software tools for designing electronic systems such as printed circuit boards and Integrated Circuits (IC).

Since EDA software runs a long time in batch mode, the generated log files by design tools are often the user's sole feedback. Users constantly need to check progress by listing these logs. Analysing and understanding these log files design is a daunting task. Design verification is also the process of going through each stage of a design and ensuring that it will do what the specification requires it to do. Here, users also need to look for information in verification logs to evaluate the produced IC.

Design-quality monitoring and reporting has now become a discipline in itself. It can make the difference between meeting delivery schedules and not meeting them, between one-pass silicon and expensive respins, and between meeting a market window and missing it entirely. Thus, an automatic and efficient solution to verify the design quality based on the information contained in the log files is an essential requirement.

In this domain, to ensure the design quality, there are some quality check rules which should be verified. These quality check rules are usually formulated in the form of natural language questions (eg., "*Capture the total fixed cell STD*" or "*Captures the maximum Resistance value*"). Verification of these rules is principally performed by analysing the generated log files. In the case of large designs that the design tools may generate megabytes or gigabytes of log files each day, the problem is to wade through all of this data *to locate the critical information* we need to verify the quality check rules.

These log files typically include a substantial amount of data. Accordingly, manually locating information is a tedious and cumbersome process. A wide array of techniques has been developed to help in the retrieval of relevant information from the log files. Unfortunately, the large amount of log data that must be analysed may overwhelm the presently available techniques, thereby resulting in a time-consuming and often error-prone process. This may be especially problematic in systems that handle high volumes of log data.

Furthermore, the particular characteristics of log files, specially those generated by EDA design tools, rise significant challenges in retrieval of information from the log files. The specific features of log files limit the usefulness of manual analysis techniques and static methods. Automated analysis of such logs is complex due to their heterogeneous and evolving structures and the large non-fixed vocabulary. Since the specificities of the log files is a primordial issue which requires to be developed, we devote Section 1.3 to discuss this point.

In the current systems, Information extraction on log files is typically done by manually-created regular expressions. But it is is time-consuming and error-prone. Moreover, these patterns are not flexible to the structure or vocabulary changes, which is frequently occurs in log files. Changing the design tool or even updating to a new version can results into a considerable change in vocabulary and structure of the corresponding generated log files. Creating the regular expression patterns also needs to locate and find the seeking information in log files. Beside being time-consuming and error-prone task, it needs a specialized knowledge about the structure and vocabulary of all types of log files.

Although information extraction in log files generated by IC design tools is attractive for automatic design management, monitoring and design quality verification, are not systematically exploited in an efficient way. Automatically locating information in huge log files can significantly help these domain engineers to understand and analysis the data contained in log files. Moreover, by automatically locating a requested information in log files, we do not need any more to build the complex and sophisticated extraction patterns which are used to avoid the extraction of structurally similar information.

Our research on locating requested information in log files is involved in the domains of Information Retrieval (IR), Natural Language Processing (NLP), and Question Answering Systems (QAS).

Information Retrieval & Question Answering Systems

Information retrieval aims to find documents related to a topic specified by a user. The topic is normally expressed as a list of specific terms. However, the needs of some application domains make information retrieval methods inefficient. Indeed, when the goal is to find specific and concise answers, information retrieval systems are not relevant due to the considerable number of documents that they retrieve as possibilities. Moreover, the information found in retrieved documents is not always correlated with the queries. That is why Question Answering (QA) systems are an important research topic nowadays. Question answering systems aim to find a relevant fragment of a document which could be regarded as the best possible concise answer for a question given by a user.

Natural language processing is a common field of computer science and linguistic which exploits how to computationally understand human natural languages. Ralph Grishman presents in [Grishman, 1984] two primary roles of NLP in retrieval of large bodies of information which are providing an interface to information retrieval systems, and automatically structuring texts so that their information can be more easily processed and retrieved. NLP has a special role in computer science because many aspects of the field deal with linguistic features of computation and NLP seeks to model language computationally [Joshi, 1991]. NLP is largely used in IR and QA systems to enhance the retrieval performance by providing semantic, statistic, and syntactic linguistic knowledge. Application of NLP methods largely depends on the type of textual data and its characteristics.

According to the question types and kind of resources documents, there are two main categories of QA systems: (1) Open domain and (2) Restricted domain.

Open domain QA systems try to answer questions seen in general domains. They deal with general information, which is usually large corpora consisting of documents of several general fields (e.g. corpus of web pages). Open domain QA systems have been evaluated since 1999 in TREC[1] (Text REtrieval Conference) American evaluation

1. http://trec.nist.gov/

campaigns. Contrary to open domain QA systems, *restricted domain* QA systems are designed to answer questions in a *specific* area.

In this kind of QA systems, information resources are technical and specific documents, and specific or technical questions are dealt with. Restricted domains, also called closed domains, have certain characteristics which make the methods of open domain QA less useful [Doan-Nguyen and Kosseim, 2004].

In this work, the information that we look for are described by the quality check questions. In order to locate information in log files, we are mainly interested in passage retrieval phase of question answering systems. Passage retrieval is the task of searching for passages which may contain the answer for a given question.

In this thesis, we are looking to propose a complete solution to locate information in a special kind of textual data, i.e., log files generated by EDA design tools. The contributions of this thesis are motivated by the challenges raised in locating information in log files because of their specificities explained in Section 1.3.

1.3 Data Description

This section is dedicated to the description of data (i.e., log files generated by EDA [2] tools) that we deal with in this work. First, we start by presenting different kinds of log files addressed in scientific literatures. Second, we give a brief introduction of EDA tools and subsequently we present the log files that they generate. Focusing in this work on EDA tool log files, we continue by developing the particularities of these "textual documents". Since most of challenges in this domain are due to the specificities of log files, understanding the characteristics of these log files can help to better identify and know the difficulties in locating information in this domain.

2. Electronic Design Automation.

```
fcrawler.looksmart.com - [26/Apr/2000:00:00:12 -0400] "GET /contacts.html
fcrawler.looksmart.com - [26/Apr/2000:00:17:19 -0400] "GET /news/news.html
111.111.111.111 - [26/Apr/2000:00:23:48 -0400] "GET /pics/wpaper.gif
111.111.111.111 - [26/Apr/2000:00:23:48 -0400] "GET /pics/5star2000.gif
111.111.111.111 - [26/Apr/2000:00:23:50 -0400] "GET /pics/5star.gif
111.111.111.111 - [26/Apr/2000:00:23:51 -0400] "GET /pics/a2hlogo.jpg
```

Figure 1.1: A fragment from the server logs for JafSoft Limited.

1.3.1 Presentation of different log files

We describe below three main types of log files: Transaction log files, execution log file, and reporting log files.

Transaction log files. Data collected on *communications* between a system and its users are usually registered in transaction log files. Transaction log files are usually considered as data collection resources which are used to register information regarding the type, content or time of *transactions* [Rice and Borgman, 1983]. As J. Valdman notes in [Valdman, 2001-04], it seems that the most developed area of transaction log file analysis is the WWW[3] industry. Transaction log analysis (TLA) typically addresses either issues of system performance, information structure, or measurements of user interactions [Jansen, 2009]. In this area, we can note Web server log files which register data regarding user access to Web servers. These log files are largely exploited in research on Intrusion Detection or Pattern Extraction [Yamanishi and Maruyama, 2005], [Facca and Lanzi, 2005]. In this category, we also have Web searching log files. Web search logs are electronic records of interactions between a Web search engine and users searching for information.

Figure 1.1 shows a fragment from the server logs for the JafSoft Limited site (http://www.jafsoft.com/). This fragment, giving information about user having visited the site, indicates, for example, two visits from looksmart.com which have occurred the 26 April 2000 for contact and news pages. Regarding transaction log file structure, we

3. World Wide Web

can note that they have usually some points in common. For example, *each line* in the file represents a *single* "hit" or in other world a *single transaction* which consists of a number of usual fields like "user identifier", "timestamps", "query type", and "query result code".

Execution log files. Generation of log files by computational applications is also usual. These log files, usually called execution log files, contain information about different *statuses* of a system or an application and *changes* happened in that system.

```
2011-03-15 23:32:30.067 iPartition Demo[175:207] Successfully installed authorization right.
iPartition Helper: need to self-repair.
iPartition Helper: self-repair complete.
2011-03-15 23:32:44 iPartition Demo: Warning: MBR length (83GB) differs from GUID length
(42GB)
2011-03-15 23:32:44 iPartition Demo: HFS error: HFS: dirty volume opened read-only
2011-03-15 23:32:47 iPartition Demo: Warning: MBR length (83GB) differs from GUID length
(42GB)
```

Figure 1.2: A fragment of iPartition application execution log file.

Execution logs are generated by instrumenting or monitoring an application. These log files are widely available and helpful in monitoring, remote issue resolution, and system understanding of complex enterprise applications. Moreover, the availability of execution logs continues to increase at a rapid rate due to legal acts like Sarbanes-Oxley Act of 2002 (http://www.soxlaw.com/) which stipulates that the execution of telecommunication and financial applications must be logged [Jiang et al., 2008].

Although *execution logs* may not follow a strict format, they have *one general structure*: A log line is a mixture of static and dynamic information on a single event. Figure 1.2 illustrates few lines of the iPartition application execution log file. These lines give tracing information on application status.

In the category of application execution log files, we can also include log files generated by diagnostic imaging systems. Indeed, these systems may also be configured to generate one or more log files [Thattil, 2008]. The log files may include functions and activities performed by the imaging system, often in a time-associated format. Accordingly, these log files may be used by technicians to facilitate detection of faults associated with the diagnostic imaging system and subsequent diagnosis and/or servicing.

Report log files. Here we introduce a kind of log files that we call report log file. Although all types of log files report information, the source and nature of information may differ depending on the type of log files. Information reported by a system transaction or execution log files are resulted by occurrence of events which concern that system (e.g., user interactions or system status changes). However, some computational systems give textual reports on characteristics of other elements which are produced by the systems.

To exemplify, we can mention log files generated by CAD programs. CAD refers to Computer-Aided Design applications which are used in design process, virtual test and design documentation. In design verification process, CAD systems may generate log files reporting the design characteristics and result of tests virtually performed on the design. Data in these log files do not concern the system itself, but rather its product characteristics and quality.

Beside CAD systems, we have also verification systems or test platforms which generate test results in format of log files. In this category, we can also mention log files generated by EDA systems. These log files are digital reports on Integrated Circuits (IC) design configurations, conditions, and performed verification tests on designed IC. In this work, we will mainly focus on this type of log files. More precisely, we aim at retrieving relevant information in EDA application log files to assist design quality maintenance process. We dedicate the following section to the description of EDA log file features.

1.3.2 EDA log file Features

To clarify the specificities of EDA log files, we illustrate in Figures 1.3, 1.4, and 1.5 two fragments of two log files log_A and log_B generated by two different EDA tools so-called here respectively $Tool_A$ and $Tool_B$[4]. We consider log files[5] as a kind of "**complex textual data**", i.e., containing *multi-source*, *heterogeneous*, and *multi-format*

4. According to this thesis confidential terms, we cannot provide the real name of the log files. However, all examples are obtained from real industrial data.

5. In this thesis, we simply use "log files" or "log" to refer to EDA tool log files. Any other kinds of log files are explicitly mentioned by their type.

```
Policy: DESIGN  Ruleset: RESETS
          <violated>/<checked> x <label> [<severity>]:<message>
          ---------- ------------------- ----------------------
          0/1 x NTL_RST01 [ERROR]: "Use only one reset domain"
          0/1 x NTL_RST03 [ERROR]: "All registers must be asynchronously set or reset"
          0/1 x NTL_RST04 [ERROR]: "A reset signal is not allowed to be used as an input to control path logic"
          0/1 x NTL_RST05 [WARNING]: "Don't use asynchronous set/reset signal except for initial reset"
          0/1 x NTL_RST07 [WARNING]: "Don't use one reset signal for asynchronous reset"
          0/1 x NTL_RST09 [ERROR]: "Reset signal gated with an OR gate"
          0/1 x NTL_RST10 [ERROR]: "Reset signal gated with an AND gate"
          0/1 x NTL_RST13 [WARNING]: "Inverter on reset path detected"
          0/1 x NTL_RST14 [ERROR]: "Reset pin not connected to reset net"
          0/1 x NTL_RST16 [ERROR]: "Reconvergent path on reset tree detected"
          0/1 x NTL_RST17 [ERROR]: "Reset gating must take care of the flip-flop triggering edge.
          0/1 x NTL_RST18 [ERROR]: "Reset signal must not interact with the other latch pins"

Policy "LEDA", Ruleset "RESETS":
          <violated>/<checked> x <label> [<severity>]:<message>
          --------------------   ------- ---------------------
          0/1 x B_1400 [WARNING]: "Asynchronous reset/set/load signal
          0/1 x B_1401 [WARNING]: "Synchronous reset/set/load signal
          0/1 x B_1403 [ERROR]: "Flip-flop assigned but not initialized"
          0/1 x B_1404 [WARNING]: "Asynchronous reset/set/load <%item> exists in module/unit"
          0/1 x B_1405 [WARNING]: "<%value> asynchronous resets in this unit detected"
          0/1 x B_1406 [WARNING]: "<%value> synchronous resets in this unit detected"
          0/1 x B_1407 [ERROR]: "Do not use active high asynchronous reset/set/load"
          0/1 x B_1408 [ERROR]: "Do not use active high synchronous reset/set/load"
          0/1 x B_1409 [WARNING]: "<%value> asynchronous resets in always/process block"
          0/1 x B_1410 [WARNING]: "<%value> synchronous resets in always/process block"
          0/1 x B_1411 [WARNING]: "<%value> asynchronous sets in this unit detected"
          0/1 x B_1412 [WARNING]: "<%value> synchronous sets in this unit detected"
          0/1 x B_1413 [WARNING]: "<%value> asynchronous sets in always/process block"
          0/1 x B_1414 [WARNING]: "<%value> synchronous sets in always/process block"
          0/1 x B_1415 [WARNING]: "<%value> asynchronous loads in this unit detected"
          0/1 x B_1416 [WARNING]: "<%value> synchronous loads in this unit detected"
          0/1 x B_1417 [WARNING]: "<%value> asynchronous loads in always/process block"
          0/1 x B_1418 [WARNING]: "<%value> synchronous loads in always/process block"

Policy: DESIGN  Ruleset: RESETS
          <violated>/<checked> x <label> [<severity>]:<message>
          ---------- ------------------- ----------------------
          0/1 x NTL_RST04 [ERROR]: "A reset signal is not allowed to be used as an input to control path logic"
          0/1 x NTL_RST05 [ERROR]: "Don't use asynchronous reset signal except for initial reset"
          0/1 x NTL_STR50 [ERROR]: "Inhibit: Latch to Latch path detected"
          0/1 x NTL_STR51 [ERROR]: "Inhibit: Latch with reset"


//        Total Module Instance Coverage Summary

               TOTAL      COVERED       PERCENT
lines            501        158            31.54
statements      501        158            31.54


```

Figure 1.3: Fragment of an EDA log file generated by $Tool_A$ (page 1)

```
run build_model
  ---------------------------------------
  Begin build model
  ---------------------------------------
  There were 1482 primitives and 4 faultable pins
  Warning: Rule B7 (undriven module output pin) was violated 1 times.
  Warning: Rule B8 (unconnected module input pin) was violated 70 times.

Test Coverage Result: Total Coverage
   Line No          Coverage    Block Type
   146                  0             VHDL_IF

------ Design Statistics:

Number of ports:            00
Number of nets:             10
Number of cells:            20

Combinational area:         40
Noncombinational area:   50

#patterns     #faults         #ATPG faults  test      process
 stored     detect/active     red/au/abort  coverage  CPU time
 ---------  -------------  ------------  --------  --------
 Begin deterministic ATPG: #uncollapsed_faults=28650, abort_limit=10...
 32         26155   2495        0/0/0    53.50%      0.08
 50          2141    353         1/0/0   57.01%     0.11
 77           219    129          4/2/9    57.37%     0.21

Total IO Pad Cell Area                          : 1076208.64

------ Design Statistics:

Number of Instances                             : 13628
Number of Nets                                  : 14293
Maximum number of Pins in Net                   : 531

     IO Port summary
Number of Primary I/O Ports                     : 388
Number of Input Ports                           : 259

====================================================
|| DesignWare Building Block Library       ||     Version       || Available        ||
====================================================
|| Basic DW Building Blocks                 || Y-2006.06-DWBB_0606 ||    *  ||
|| Licensed DW Building Blocks              ||                  ||                  ||
=====================================================

Std cell utilization:          0.93%   (449250/(48260400-0))
Chip area:                     48260400 sites, bbox (210.00 210.00 4140.00 4139.60) um
Std cell area:                 449250     sites, (non-fixed:449250 fixed:0)
                                31625       cells, (non-fixed:12345   fixed:130)
Macro cell area:               0             sites
```

Figure 1.4: Fragment of an EDA log file generated by $Tool_A$ (page 2)

```
Instance        Cells   Cell Area   Net Area
------------------------------------------------
aes_cipher_top  14290    1138155     437930
  u0              2964     250596      71024
  us21            616      39185       14694


Timing
------
Warning : Possible timing problems have been detected in this design.

1
   set_top i:/WORK/fifo
Setting top design to 'i:/WORK/fifo'
Status:  Implementing inferred operators
1
#Creates the clock
create clock period $Clock_PERIOD -name $Clock_Name [get_ports $Clock_PORT]

Top design successfully set to 'i:/WORK/fifo'
Implementation design set to 'i:/WORK/fifo'


 Slack     Endpoint   Cost Group
-------------------------------------
   +XXps sa22_reg[7]/D default
   -XXps sa22_reg[7]/D default

(halcheck): W,CMBPAU (TestFail): Combinational path detected through '%s' in module/design-unit '
(halcheck): W,ASNCFL (TestFail): Asynchronous feedback loop detected through set/reset of flip-flop(s)
(halcheck): W,MULMCK (TestFail): Clock '%s' for flip-flop '%s' is derived from master input '%s'
(halcheck): W,GTDCLK (TestFail): Clock gating detected for clock '%s' of flip-flop '%s'
(halcheck): W,MRSTDT (TestFail):  Synchronous set/reset detected in '%s' and asynchronous set/reset detected
(halcheck): W,LATINF (TestFail): Latch inferred for '%s'
(halcheck): W,DIFRST (TestFail): Set/Reset '%s' is being renamed to '%s'

                         Coverage Summary Report, Module-Based
             ================================================

B = Block, BR = Branch, E = Expression
C = Cumulative coverage for that coverage type including sub hierarchy

B:   12.34% (24/26)        BC:   56.78% (685/688)
E:   90.12% (19/32)        EC:   34.56% (502/710)

Clock              Period   Waveform          Attrs         Sources
------------------------------------------------------------------------------
CLK                12345.67890   {0 2}                    {clock}
CLK_tmp            98765.43210   {5 6}                    {clock}


****** Signal Coverage ******
   CV_INT[0:255]                      No    No
   ALG_DATA_INT[0:127]                No    No
   END_KEY                            No    No

*** Clustered FPlan Seeds ***
# of standard cell seeds is: 0
# of soft standard  module seeds is: 19
# of instance cell group seeds is: 8

**** MODULE, INSTANCE ****
MODULE TB_ECB_VK_DEC_ITER.TOP_LEVEL.CTRL
FILE  /users/AES/src/controller_iter.vhdl

*** Clustered Preplaced Objects ***
# of preplaced io is: 18
# of preplaced hard macro is: 0
# of preplaced standard cell is: 24678
```

Figure 1.5: Fragment of an EDA log file generated by $Tool_B$

data. In addition, design tools change over time, often unexpectedly. Therefore, the *format of the data* in the log files changes, which makes automatic data management difficult. In this domain, we deal with a huge corpus of logs. For instance, size of a single log file can simply achieves more than 1 GB. We present below the main characteristics of these log files and compare them with classic texts written in natural language.

Multi Sources Data. In EDA domain, *several design tools* can be used. Although logs of the same design level contain the *same* information, their *structures* and *vocabulary* can vary significantly *depending* on the *EDA design tool used*. For instance, in log_A (see Figure 1.3, page 14), the lines 48 to 52 report information about "`Statement coverage`" in a design. The *same information* in log_B (see Figure 1.5, page 16) is expressed in lines 37 to 44 with *totally dissimilar structures* and almost *different vocabulary*.

Although log files are semi-structured data and should conform to a grammar, acquisition or knowing of their generator grammar is challenging and in some situations irrelevant. Indeed, according to some constraints like data confidentiality or software licensing, access to design tools is not always possible. Moreover, extracting language model or grammar of log files generated by a given tool requires sufficient amount of data (i.e., logs) generated in different configurations by that tool. Unfortunately, creation of such corpus is tedious, time/resource consuming and sometimes not feasible according to the available number of log files. Furthermore, design tools are susceptible to change and evolve in time, which means that the language model used to generate their logs may change too. Note that, in this domain, a user can also supply self generated log files which do not exist officially and thus cannot be previously seen by an IE system.

In such conditions, learning grammar of log file generators and extraction of their language model is revealed irrelevant and not efficient in term of performance. That is why we need solutions which enable to analyse log files despite their heterogeneous structures and with no need to know their language model. These methods also have to be flexible or adaptable to changes and evolutions in log file generators.

In this work, we are initially faced with the question that whether the NLP classic methods are capable to process such a multi-source data. In other words, we wonder if multi-source and multi-vocabulary aspects are relevantly investigated in classic NLP methods. Actually, in processing classic texts, we do not usually deal with multi-structure, multi-vocabulary data. An information can be rephrased differently, but its vocabulary either does not change or we have lexico-semantic variations of terms which can be recognized by the use of NLP and semantic resources. For instance, there are some researches about the processing of paraphrasing based on Morpho-syntactic variants or Inference [Rinaldi et al., 2003]. As illustration, two below sentences are an instance of Morpho-syntactic variant paraphrase:

- *Oswald killed Kennedy. / Kennedy was killed by Oswald.*
- *Edison invented the light bulb. / Edison's invention of the light bulb.*

As shown, we have different rephrasing, but always the same essential keywords in two sentences. Whereas the vocabulary changes in log files are not due to lexico-semantic variation of terms, but usually different instance/notion nomination or standardization. Moreover, there is not any relevant domain dictionary or semantic resource available in the context of EDA.

Data Heterogeneity. Data *heterogeneity* occurs not only between the log files produced by different tools, but also within a given log file. For example, the symbols used to present an object, such as the header of tables, change in a given log. For instance, in log_A (see Figure 1.4), lines 76-78 and lines 96-98 present two different kinds of table header. Moreover, the same symbol used in the line 78 to present a table header is also used in lines 3 or 19 within data blocks. Similarly, there are heterogeneous kinds of punctuation, data formatting, and data representation. Whereas we have a normalized punctuation and data representation format in every natural language.

Language. Although the language used in these logs is English, their contents do not usually comply with "*conventional*" grammar. We also deal with a specific vocabulary which is not covered by conventional lexico-semantic resources. Some terms may have different signification based on the special characters preceding them. In the processing of log files, we also deal with multi-format data: textual data, numerical data, alphanumerical, and structured data (e.g., table and data block). There are also many technical words that contain special characters which are only understandable in the light of the domain expert.

Logical Structure. Recognizing log file heterogeneous structures and taking it into account is an essential point in locating information in log files. Indeed, the classic documents often have logical units like sentences, paragraphs, or sections which develop a single idea or topic. In classic text, we can exploit the elements marking the logical units, such as white lines, indentations or periods. But these conventional logical units are not significant in log files. Instead of such structures, we deal with complex and heterogeneous logical units like tables, data blocks, and specific strings *marking* the beginning of new information. To exemplify, in log_A (see Figure 1.4, page 15), the lines 76 to 82 correspond to a "data table" logical unit while we also have another form of "data table" logical unit illustrated in lines 96 to 101. These textual structures, more complex than conventional structures, require most advanced methods to be relevantly recognized in all types of log files. Otherwise, the irrelevant identification of boundaries of logical units in log files could split correlated information into several different segments. This issue can bias the results of information extraction in log files.

Due to these specific characteristics of log files, NLP and IE methods, developed for texts written in natural language, are not necessarily well adapted to log files. We note that the particularities of these log files exist in most types of log files regardless of their application domain. These particularities raise challenges that motivated our work.

EDA Log Files vs. Other Mentioned Log Files. The previous mentioned log files share some characteristics with EDA log files. For example, they also contains numerical data and not structured natural language phrases. In despite of these common characteristics, EDA log files differ in many aspects from other kinds of log files.

Most of the characteristics, previously mentioned for EDA log files, do not exist in other kinds of log files. Modeling the behaviour of a web application user or the system troubleshooting are the domains wherein we mainly deal with transaction and execution log files. These log files are thus largely exploited according to their application domains. The wide application of these log files has meant that today there are several standards in the producing and analysing of these kind of log files. Therefore, these log files are more homogeneous and comply with some predefined structures. For example, a transaction log file should contains the date, the time, the user identifier, and the destination of a transaction whatever is the type of server. Nevertheless, in EDA log files, as previously mentioned, we deal with a considerable heterogeneity in terms of structure and vocabulary used in log files.

Moreover, Transaction and execution log files usually do not contain complex structures like as tables or data blocks. In an transaction log file, an information about a given transaction is only contained in one line. However, in EDA log files, an information can be located in several lines or within a table. These point can significantly impact the information locating in EDA log files from the text segmentation and passage retrieval point of view.

1.4 Contributions of this Thesis

In this thesis, we present contributions to the challenging issues which are encountered in question answering and locating information in complex textual data, like log files. By each contribution, we answer to questions raised in this work due to the data specificities or domain requirements. We investigate throughout this work the main concern "*how the specificities of log files can influence the information extraction and natural language processing methods?*". Log file specificities, explained in Section 1.3.2, make classic IE and NLP methods unsuited. The problem is highlighted when we also have to take their *scalable structures* and their *special vocabulary* into account. In this context, a key challenge is to provide approaches that take the log file specificities into account while considering the issues which are specific to QA in restricted domains. We present different contributions for each level of Question Answering and Information Extraction from log files.

• **How to recognize logical structures of log files to perform text segmentation?**
In order to reduce search space and better locate information, we look for splitting the log files into relevant segments. For example, in log_A (see Figure 1.3), the text chunk

situated in lines 39 to 45 corresponds to a relevant segment which includes correlated information. We consider the logical units of log files as guidelines for segmentation. Since the conventional logical units of textual documents (e.g. paragraph or sentence) are not significant in log files, we thus propose a method to *characterize complex logical units* found in log files according to *their syntactic characteristics*. Indeed, we aim at proposing an approach to *recognize logical divisions* existing in log files. We characterize the logical units of log files by a number of “features" where features are the syntactic characteristics of logical divisions. In order to perform the feature acquisition, we propose two different methods: (i) Semi-automatic and (ii) Automatic. In the semi-automatic way, we define the features according to some heuristics based on expert knowledge. In the automatic one, we propose an original type of feature, called “*generalized vs-grams*" which allow to model the textual structures (the layouts and the composition of letters and special characters) of a document while being insensitive to the contents of the latter.

Using obtained features and a supervised classification method, we make it possible to recognize different logical units in the log files. Although we rely on “log file" data type, our approach is applicable to all kinds of textual data that have complex syntactic structures. The findings show that *the generalized vs-grams*, introduced in this work, can be used for modelling the complex logical units of documents without the need to extract document language model or to learn its language grammar which are tedious and time/resources consuming.

• **How to enhance the answer locating in log files?**

Initial queries (i.e., question keywords) are irrelevant enough in all cases due to the complexity of log files. Moreover, the specificity of such textual data and characteristics of restricted domains significantly impact the accuracy and performance of Passage Retrieval in this context. We propose a novel query expansion approach to adapt an initial query to all types of corresponding log files and overcome the difficulties like mismatch vocabularies. Within this contribution, we aim at boosting the performance of Passage Retrieval by query expansion within a QA system on a specific domain. Our query expansion approach relies on two relevance feedback steps. In the first phase, we determine the explicit relevance feedback by *identifying* the *context of questions*. We subsequently determine expansion terms by identifying significant words *representing* the *context* of questions. The second phase consists of a novel type of pseudo relevance feedback. Our method is based on a *new term weighting function*, introduced in this work, which gives a score to terms of corpus according to their *relatedness* to the *query*. Indeed, this new measure, called TRQ (Term Relatedness to Query), aims at giving a high score to terms of a corpus which have a significant probability of existing in the relevant passages. We use TRQ measure to identify the most relevant expansion terms.

To exemplify, consider "*Existence of both asynchronous and synchronous Reset or Set?*" as a question and its keywords (i.e., synchronous, set) as a query given to an IR system. Using this initial query and the logs in Figure 1.3 on page 14, we retrieve the segment located in lines 18 to 38, though the relevant segment is located in lines 1 to 15. IR is here biased in favour of second segment due to the high occurrence frequency of keywords in this segment. After applying the first phase of our query expansion approach and identifying the context of question in log_B (see lines 29 to 35), we obtain "*detected*" and "*flop*" as expansion terms. Therefore, by means of TRQ measure, we select terms "*latch*" and "*reset signal*" as two other expansion terms. Using the expanded query, we can retrieve the relevant passage in log_A.

We also investigate how to apply our query expansion approach to documents from general domains. Within this study, we demonstrate how to improve this approach in order to meet the characteristics of open domain documents. This highlights the efficiency of our approach, initially designed for log files, in the context of open domains.

• **How to improve information extraction by means of terminological knowledge extraction?**

We observed that the domain lexical knowledge can improve the performance of information retrieval from log files. Thus, we seek to integrate a kind of lexical knowledge on the log file domain to our Information Extraction approach. In this work, we particularly aim at proposing a relevant approach to explore the lexical structure of log files in order to extract the domain terminology. The domain terms are subsequently used to enrich the log file features. This terminological knowledge enables us to better characterize log files and identify their relevant features.

We here introduce our approach, named Exterlog (EXtraction of TERminology from LOGs), to extract the terminology of log files. We study within our approach the relevance of two main methods of terminology extraction based on the extraction of co-occurrences with and without the use of syntactic patterns.

To exemplify the usage of terminological knowledge, we refer to the previous example on query expansion. As shown, one of the expansion term is "*reset signal*" which is identified as a domain relevant term by Exterlog. This two-word term is more significant and discriminatory than single-word terms.

We note that according to the particularity of such data and due to the high noise ratio in results, the result evaluation is a challenging issue in this context. To emphasize the precision of results as a must according to the accuracy of context, we have to define the noise filtering method which complies with the particularity of such data. Thus, we propose a *candidate term evaluation method* which scores terms using a statistical measure based on the Web and the context of documents. In the scoring function, we take the *context recognition* into account as a factor which can influence the evaluation of extracted terms.

1.5 Outline of this Thesis

The rest of this thesis is organized as fellow.

- In Chapter 2, we study the problem how to split log files into relevant text chunks, called segments, where each segment contains correlated information. We first present the different methods of text segmentation and the related existing work. Then we propose our solution to segment log files based on the recognition of their logical units. We introduce how to characterize the logical units by means of a new kind of grams, called vs-grams. We finalize by presenting the results of experiments.

- In Chapter 3, we study how to retrieve the relevant segments of log files, called passages, based on a given query. We mainly focus on how to improve the relevance of queries in order to better locate information in all types of log files. We first discuss the existing work in the domain of passage retrieval as well as work in the domain of query expansion. Then we propose an approach of query expansion based on two novel types of relevance feedback. We also present a new term weighting function, called TRQ (Term Relatedness to Query) which aims at giving a score to terms based on their probability to exist in the relevant passages. We subsequently discuss the application of our query expansion approach, initially designed for the domain of log files, in general domains. Then we continue by presenting the experiments performed using the real data.

- We devote Chapter 4 to present the acquisition of terminological knowledge in the log files. We first present the motivations of using the terminological knowledge in query expansion and passage retrieval. Then we study the existing work in the domain of terminology extraction. We subsequently present our approach Exterlog to extract the domain terms in the log files. We also introduce an approach to evaluate the relevance of extracted terms. We finally describes and compares the various experiments that we performed to extract terms in the log files and to evaluate the performance of Exterlog.

- Chapter 5 is devoted to the industrial application of our approach. We introduce the integration of our solution to locate information in log files into a commercial enterprise quality verification software. We first introduce the specifications of our solution. Then we demonstrate how each phase of our solution have to be executed.

- Finally, in Chapter 6, we summarize the work presented in this thesis and propose the perspectives of our future research directions.

Chapter

2

Log File Segmentation

Do not go where the path may lead, go instead where there is no path and leave a trail.

Ralph Waldo Emerson

Preamble

In this chapter, we present our study on segmenting log files into relevant text chunks. We introduce text segmentation and its application in Question Answering systems. Then, we discuss the related work about text segmentation and their relevance to the context of log files. Furthermore, we present our contribution to log file segmentation. We finally present test results and provide a detailed discussion about the presented methods and findings. The contribution presented in this chapter is also published in [Saneifar et al., 2011b]

Contents

2.1 Introduction

Passage Retrieval is a major constituent of Question Answering systems and is aimed at retrieving relevant passages in documents which contain answers to a questions. As an accurate and reliable definition, a passage is a fixed-length sequence of words which can begin and end anywhere in a document [Callan, 1994],[Kaszkiel and Zobel, 2001],[Ofoghi et al., 2006].

Passage Retrieval significantly influences the performance of QA systems because final answers are sought in the retrieved passages. Moreover, previous studies show that extracting answers in chunks of textual documents give better results than searching for information in the whole document [Llopis et al., 2002a],[Kaszkiel and Zobel, 2001]. For instance, in [Kaszkiel and Zobel, 2001], authors argue that Passage-level access has several advantages over document-level access. Passage retrieval can be considered as a means of locating information in documents.

By passage retrieval we reduce significantly the search space wherein we look for answers to questions. However, there is not a general agreement about how one should define those passages in order to obtain an optimum performance [Llopis et al., 2002a]. The main differences between different PR systems are the way that they select the passages, that is to say, what they consider as a passage? how to discover its boundaries? and how to evaluate its relevancy?

We develop the task of passage retrieval in Chapter 3. Thus, here we give a brief introduction to principles of passage retrieval in order to better describe the motivations and application of our text segmentation approach.

In most passage retrieval methods, we distinguish two main phases: (1) *passage segmentation*, (2) *passage ranking (scoring)*. Passage segmentation is the task of determining text segments in documents which are considered as candidate passages. In passage segmentation, the issue is how to define the passage boundaries? and how to recognize them in a corpus? While passage ranking assesses the relevancy of passages according to a given query. By means of scoring and measuring functions, one evaluates how much a passage is correlated to a query.

The passage retrieval methods are distinguished based on the way they treat these two issues. We here differentiate passage retrieval methods in terms of the moment when the passage segmentation is carried out. In this category, we have *index-time* passaging methods versus *search-time* passaging ones.

In index-time passaging, documents are split into text chunks (segments) before indexing them [Tiedemann and Mur, 2008]. This means that the determination of passage boundaries or in other word the cutting documents to several segments (i.e., passages) is carried out before analysing questions. Thus, passage types and their size are independent from queries. In these methods, the passage ranking is performed later by evaluating the relevance of each candidate passage according to a given query. Contrary to the index-time passaging, in search-time methods, one determines passages along with searching for answers [Llopis et al., 2002b]. In these approaches, the passage segmentation is dynamically carried out along with passage ranking based on query properties. This means that the boundaries of a passage are dynamically determined ensuring that the best relevance score is obtained for the passage. The index-time methods allow a quicker calculation; nevertheless, the second methods allow different segmentation models in accordance with query properties [Llopis et al., 2002b].

In our QA system, a corpus of log files can be used as a data source for several list of questions where each list contains in average more than one hundred questions. Moreover, we deal with huge corpus of log files. The size of a given log file can simply reaches more than 1GB. Furthermore, log files are semi-structured data which helps to relevantly determine concise text segments when each segment only treats a single topic. Thus, we choose *index-time passage segmentation* which implies that text segmentation is independent from queries. In this way, we also obtain a considerable performance gain as we do not require to perform passage segmentation for each query. Our PR approach hence requires that documents (log files) to be split into candidate passages (segments) before passage scoring and retrieval. Therefore, we perform a text segmentation in order to obtain candidate passages.

However, the specificities of log files (see Section 1.3), like the fact that they contain complex *heterogeneous* structures such as *tables*, *lists*, *data blocks* or that they do not comply with *classic text structures*, make conventional text segmentation methods irrelevant in the context of log files. In the processing of log files, we also deal with *multi-format data*: textual data, numerical data, alphanumerical, and structured data which can influence the relevance of current text segmentation methods. We will discuss these issues in Section 2.2.

In this chapter, we study the segmentation of complex textual data like log files. For this purpose, we first study the related work in the text segmentation domain. We analyse their relevance in the context of log files. This means that we look to find how the characteristics of log files can influence the classical segmentation methods. Then we study how to take the visual structure of log files into account in order to design a relevant segmentation method. Therefore, the presence of other kind of data, like numerical data or tables, does not bias or influences the accuracy and performance of our approach. Moreover, this approach is content and language independent which makes it relevant and applicable to other kinds of textual data sharing the same specificities and structures as log files.

In order to identify the visual structure of log files, we characterize the complex logical units found in them according to *their syntactic characteristics*. Recognition of these logical units is the main core of our segmentation approach. Within our approach, we also present the original notion of *generalized vs-grams* which is used to automatically extract the syntactic characteristics of special structures found in log files.

The rest of the chapter is organized as follows. A detailed study on related work and their relevance in the context of log files is discussed in Section 2.2. Then, in Section 2.3, we present the main phases of our segmentation approach which is based on the recognition of logical units. Sections 2.4 and 2.5 are devoted to the development of different phases of our approach notably the characterization of logical units. We also present the original notion of "generalized vs-grams" which is used to characterize the logical units in Section 2.5. Results of experiments are presented in Section 2.6. A discussion is addressed in Section 2.8.

2.2 Related Work

In this section we present the current work in text segmentation domain. This task is essential in domains like Information Retrieval, Question-Answering systems, Text Summarization, dialogue generation, or improving document navigation for the visually disabled [Kaszkiel and Zobel, 1997],[Sun and Chai, 2007],[Carroll, 2010],[Choi, 2000].

Text segmentation is defined as the task of splitting a text document into several coherent segments and to discover the topic boundaries [Choi, 2000]. As mentioned in [Beeferman et al., 1999], even tough this task may seem a seemingly simple problem, but it actually proved quite difficult to automate.

We detail here text segmentation methodologies which are used to split a document into candidate passages. According to [Kaszkiel and Zobel, 2001] and [Callan, 1994], we have three types of passages: *semantic* (thematic), *window* passages, and *discourse*. Semantic segmentation consists in identifying various topics conveyed by the text, to segment it into homogeneous units forming thematic blocks [Tarek, 2003],[Ponte and Croft, 1997b]. The semantic passages correspond to the thematic structure of documents [Kaszkiel and Zobel, 2001]. In the case of "discourse passages", the segmentation is carried out based on logical divisions or in other words, the discourse units in documents. Indeed, the documents often have logical (discourse) units like sentences, paragraphs, or sections. The logical components of documents can be regarded as passages (segments) [Kaszkiel and Zobel, 2001],[Callan, 1994]. Finally, it is also possible to segment documents based on a sliding phrase or a line window. The choice of segmentation type depends largely on the objectives, application domain, and specially document characteristics.

2.2.1 Semantic segmentation

Semantic passages are based on topics derived by segmenting documents into single-topic units. This means that a document is split into semantic pieces, according to the different topics in the document [Llopis et al., 2002a],[Kaszkiel and Zobel, 2001],[Ponte and Croft, 1997a]. The semantic segmentation methods principally rely on the usage of word frequency, lexical co-occurrences, or lexico-semantic relations to identify subject changes in documents.

TextTiling is a well-known semantic segmentation approach which uses word frequencies to recognize topic shifts [Hearst, 1997]. In TextTiling, the main cues for identifying major subtopic shifts are patterns of lexical co-occurrence and distribution. Marti A. Hearst assumes in [Hearst, 1997] that a set of lexical items is used during the course of a given subtopic discussion, and when that subtopic changes, a significant proportion of the vocabulary changes as well. He aims at splitting texts into continuous, non-overlapping multi-paragraph subtopic segments. Hearst notes that TextTiling is mainly defined to apply to texts that are not heavily stylized or structured. The topic change is measured based on lexical co-occurrence patterns. Measurement scores are calculated at sentence-size unit level. Hearst also proposes to compare the text block (composed of a fixed size sentences) to each other based on a lexical score which relies on the number of common terms. TextTiling being a well-known semantic segmentation, we evaluate its application in the context of log files. This evaluation, presented in Section 2.7, gives a survey about the efficiency and relevance of semantic segmentation approaches in the context of textual data like log files.

The word repetition in a sentence and redundant utterances are considered as a lexical cohesion measure in [Walker, 1992]. The value of this indicator is used to find the segmentation points while it can have various values for different types of texts. The notion of lexical chain is studied in [Kan et al., 1998] to identify topic changes in a document. A lexical chain is defined as a series of words which links the occurrences of a given term in a document. The chain size is determined (limited) by the number of sentences between two occurrences.

[Kozima, 1993] proposes to calculate a lexical cohesion profile which locates segment boundaries in a text. This approach relies on the similarity of words in a sentence. Word similarity, representing word cohesiveness, is calculated using a semantic network. This semantic network is constructed from an English dictionary.

A text segmentation method principally based on unsupervised machine learning is presented in [Caillet et al., 2004]. In this work, each segment is represented in a reduced dimensional space called concept space which is richer than the bag of words representation. A concept is defined as a group of representative terms. The extraction of concepts is performed according to the co-occurrence of words in paragraphs. Once concepts are discovered in a text, this system splits texts into coherent paragraphs. These paragraphs are characterized in the concept dimensional space, using an unsupervised classification. They finally create different semantic segments by combining the paragraphs which are "semantically" close to the meaning of each concepts. This means that the non-adjacent segments can be combined into a single theme corresponding to a topic. The system provides a collection of segments labelled according to their concept.

There are also semantic segmentation methods calculating a semantic distance between terms. They also take term disambiguation methods (e.g. synonymity discovery) into account to better calculate semantic relations between terms. These methods often rely on external resources such as domain ontologies or semantic networks [Labadié and Prince, 2008].

Studying main researches regarding semantic segmentation shows that topic change discovery mainly relies on the assumption of cohesion, which is a device for making connections between parts of the text [Nguyen et al., 2011]. As it is also mentioned in [Nguyen et al., 2011], we note that the current cohesion-based text segmentation approaches focus on either term iterations, term semantic relations, or both. Co-occurrence terms, lexical chain discovery or term repetition based methods can all be classified into the category of methods which suppose that analysing term occurrences can guide to discover topic changes. Techniques based on semantic distance, document similarity using IR measures, or data mining are more sophisticated. They include relations between words that tend to co-occur in the same contexts, which have systematic and the non-systematic semantic relations [Nguyen et al., 2011].

In our domain, these methods suffer from some common points. We try to explain below why these semantic segmentation methods are not suited to segment log files.

First, the notion of subject change based on the lexical cohesion assumption is questionable in our context. Semantic segmentation methods use phrase or paragraph as the basic unit. Whereas log file, as explained above, are not natural language texts which means that notion of phrase or paragraph is not significant in this context. Furthermore, we show trough few examples that a change of co-occurrences or a change of term repetitions does not necessarily result in topic change in log files. This throws doubt on the relevance of assumptions based on term iteration or term occurrences which are used in above mentioned methods. We refer to the example log_A, presented in Figure 1.4 on page 15. Supposing lines 67 to 74 (shown below) as a text segment.

```
67 | —— Design Statistics:
68 |
69 | Number of ports:          00
70 | Number of nets:           10
71 | Number of cells:          20
72 |
73 | Combinational area:       40
74 | Noncombinational area:    50
```

After a lexical co-occurrences and repetition analysis of this segment, we discover a co-occurrence change after line 72. Based on the above assumptions, we should consider lines 69 to 71 as a segment and lines 73 to 74 as another one which treat different topics thought the entire of this chunk should be considered as a single segment according to a domain expert. Indeed, the whole chunk from line 67 to 74 reports information about "design statistics". This issue is highlighted when we are dealing with tables or numerical data in log files.

In the same log file (i.e., log_A), we have a table within lines 76 to 82 which is illustrated in the following lines.

```
76 | #patterns   #faults      #ATPG faults     test        process
77 | stored    detect/active   red/au/abort   coverage    CPU time
78 | -------    ----------      ---------      ------      ------
79 | Begin deterministic ATPG: #uncollapsed_faults=2865, abort_limit=10
80 | 32          26155 2495         0/0/0       53.50%        0.08
81 | 50            2141 353         1/0/0       57.01%        0.11
82 | 77             219 129         4/2/9       57.37%        0.21
```

A considerable vocabulary change is noticeable after line 79 as we have only numerical data. However, this change does not mean a topic change. A table has to be always considered as a single segment in log files.

Beside tables, we have many adjacent text chunks in log files which contain principally numerical data. Discovery topic changes in text chunks treating principally numerical data is not feasible by semantic segmentation methods which rely on vocabulary change or lexical cohesion. General purpose semantic resources are not relevant in specific and technical domains as they do not cover specific terms related to a specialized domain. That is why, there are specialized semantic resources in some technical domains like Medicine or Biology.

Regarding EDA domain, we do not have semantic resources and creation of such resources which enable to inference the semantic relations, is revealed time-consuming task, which also needs a lot of domain expert interventions. Thus, we are faced with some difficulties in our context in using the methods based on semantic network like ontologies.

Note that we are not looking to cluster log file text chunks which treat a same topic. That is why the methods which create segments by generating sentences [Buscaldi et al., 2010] or clustering similar text chunks [Caillet et al., 2004] are not applicable in our context regardless the fact they suffer from above issues too.

2.2.2 Window Based Segmentation

Other segmentation methodology is based on the notion of sliding window [Kaszkiel and Zobel, 1997]. As mentioned in [Tiedemann and Mur, 2008], windows can be defined in terms of words [Kaszkiel and Zobel, 2001] or in terms of sentences or paragraphs [Zobel et al., 1995]. Window segmentation imposes a single model of text division. The fixed or variable size windows may be either non-overlapping or overlapping sliding windows [Llopis et al., 2002b],[Kaszkiel and Zobel, 2001]. The window segmentation methods are also used along with passage scoring methods. In such case, windows can start with a keyword of query and continue according to the presence of other keywords. In order to find the best segments, one slides the window or changes its size and calculates the relevance of new obtained segment based on the new position of the window.

Window models have the main advantage to be simpler to accomplish [Llopis et al., 2002a]. A drawback of these methods is the potential cutting of a relevant chunk of a document into several segments, whereas it should be seen as a single segment. This error, also noted in [Callan, 1994], can simply occur in the case of tables or data block in log files. For example, in log_B (see lines 37 - 44 in Figure 1.5, page 16) we have a data block which is shown below.

```
37 |         Coverage Summary Report, Module-Based
38 |     ==================================
39 |
40 | B = Block, BR = Branch, E = Expression
41 | C = Cumulative coverage for that coverage type including sub hierarchy
42 |
43 | B: 12.34% (24/26)      BC: 56.78% (685/688)
44 | E: 90.12% (19/32)      EC: 34.56% (502/710)
```

Using a window segmentation where the window size is for example equal to 3 lines, we split this data block into three segments, though it should be treated as an unique segment.

Segmentation based on windows also ignores the structure of log files although log file structures contain relevant information to identify segments or extract information. Finally, discourse-based and semantic models have the main advantage that they return logic and coherent fragments of the document, which is quite important in Question Answering [Llopis et al., 2002a].

2.2.3 Discourse Passages

In the case of "discourse passages", the segmentation is carried out according to logical divisions or in other words, the logical units of documents. Indeed, texts are often composed of logical units like sentences, paragraphs, or sections. The logical (or *discourse*) units of documents can be regarded as candidate passages (segments) [Kaszkiel and Zobel, 2001; Callan, 1994; Salton et al., 1993]. The segment definition is intuitive in this approach, because sentences should develop a single idea; paragraphs should address a single topic, and sections are the subject of a question [Callan, 1994; Kaszkiel and Zobel, 2001]. As mentioned in [Hernault et al., 2010], in the last twenty years, the study of discourse has received continuous attention from the Natural Language Processing community. Discourse structure is fundamental to many text-based applications, such as Question Answering [Sun and Chai, 2007].

In discourse segmentation we can consider different granularities. Discourse segmentation for Information Retrieval or Question Answering is usually performed at the granularity level, distinguished in [Marcu, 2000], which consists on clause, sentence, paragraph, and section. The finer granularities are usually studied in Logical Parser studies to create logical structure of a document.

There are some solutions to identify classic logical units such as paragraphs in texts. We can also exploit the elements marking the logical units (i.e., logical divisions) such as "white lines" or "indentations" at the beginning of paragraphs. Deborah Schiffrin defines discourse markers in [Schiffrin, 1987] as a set of linguistic expressions that brackets units of talk. Kaszkiel and Zobel note in [Kaszkiel and Zobel, 2001] that existing mark-up or segmentation heuristics (such as empty lines) can be used to detect these discourse units. Using a set of corpus-specific heuristics is proposed in [Callan, 1994] in order to explicitly identify boundaries between paragraphs in TIPSTER[1] corpora.

There are also passage retrieval approaches which simply consider sentence logical unit as the base of segmentation. For example, a passage is defined as one or more adjacent sentences in AskHERMES medical QA system [Cao et al., 2011]. AskHERMES is based on a search-time passage retrieval. That means for a question, a passage is few adjacent sentences which every sentence incorporates one or more terms from the question. The choice of sentences, therefore the creation of passages, relies on measures like the total number of query terms that appear in sentences. A similar approach is also used in [Embarek, 2008] which considers sentence logical unit as the base of segmentation. These approaches can be considered as a kind of window segmentation where the window is defined as one or more sentences. These approaches are not thus suited to be used in log file domain for the same reasons.

As noted in [Llopis et al., 2002a], the discourse-based models look more effective since they are using the structure of the document itself. Similarly, Callan notes in [Callan, 1994] that discourse passages are expected to be the most effective, because discourse boundaries organize material by content. However, they also note several drawbacks and limits of discourse-based segmentation.

First, segmentation results could depend on the writing style of the document author [Llopis et al., 2002a]. Discourse passages require more consistency from writers than do semantic or window passages. If writers are sloppy or rushed then segmenting documents by textual discourse boundaries may be inappropriate [Callan, 1994].

Second, even though most documents are supplied with their structure, manual processing is required for those without it, thus making discourse passages impractical [Kaszkiel and Zobel, 2001].

Third, logical unit markers (i.e., logical divisions) are not always available or ambigu-

1. http://www.nist.gov/tac/data/data_desc.html

ous with other types of separators [Tiedemann and Mur, 2008]. For example, *headers*, *list elements* or *table rows* might be separated in the same way as discourse related paragraphs (for example using an *empty line*). Problems with the classical discourse passage approaches often arise with special structures such as headers, lists, and tables which are easily mixed with other units such as proper paragraphs [Tiedemann and Mur, 2008]. Precisely, we overcome this problem by means of our proposed segmentation approach.

In our context, data are well structured in the log files. There are, for example, structures such as tables, data blocks, and specific strings *marking* the beginning of new information. Moreover, log files are generated by computing systems which means their structural organization is based on a defined grammar. Even though we do not have access to these grammars or they vary from a log file type to another type, but it guarantees a consistency in writing log files. Therefore, the first and second mentioned drawbacks of discourse segmentation, regarding writing style and existence of document structure, do not exist in the context of log file. However, the third problem is emphasised in log files where conventional logical divisions are meaningless (e.g., paragraph indentions) or ambiguous (e.g., empty lines).

For example, the following lines demonstrate an unique segment in log_A (see lines 67 to 74 in Figure 1.4, page 15). In this segment, there are two empty lines which do not represent a logical division (i.e., they do not correspond to the beginning of a new segment).

```
67 | —— Design Statistics:
68 |
69 | Number of ports:          00
70 | Number of nets:           10
71 | Number of cells:          20
72 |
73 | Combinational area:       40
74 | Noncombinational area:    50
```

In the same time, we deal with special structures such as headers, lists and tables in log files which can take different representation formats (e.g., different table types). These textual structures, more complex than conventional structures (classic logical units), are used to gather ideas and information. Thus, we consider these complex structures (which gather the correlated information) as logical units of log files.

We hence aim at proposing an approach to recognize special logical units in log files. Our approach enables to identify different complex kinds of logical units while they can take different forms or presentation formats. By means of our approach, we overcome the third drawback of discourse-based segmentation in texts having special structures like tables, lists, or data blocks.

In the following section, we describe how to discover the special logical divisions of log files and subsequently recognize their logical units.

2.3 Global Process of Logical Units Recognition

In this section, we present our global process of identification and subsequently recognition of logical units. We introduce the main idea of the approaches used to identify complex logical units. We subsequently detail these approaches in Sections 2.4 and 2.5

Identifying different kinds of discourse units in log files requires to recognize different layouts and data presentation formats which are used to structure them. We hence look for syntactic cues which indicate a separation of consecutive text chunks. We have decided to model logical units according to their *characteristics*. This means that we determine the *syntactic characteristics* of *logical units* in log files. They allow to distinguish the beginning of a logical unit (i.e., a logical division) from the other lines in the documents. For example, in classical texts, we can consider the "*indentation*" as one of the characteristics allowing to identify a paragraph. We call these syntactic characteristics *features*. We therefore characterize the logical units of log files by defining a number of "features". In order to perform the feature acquisition, we chose two different methods: (1) semi-automatic and (2) automatic.

In the semi-automatic way, we first define the features according to some heuristics based on an expert knowledge. Afterwards, we model the logical divisions based on the presence/absence of features.

In the case of the automatic method, we propose an original type of n-grams, called "*generalized vs-grams*". The features are determined automatically by extracting generalized vs-grams in a corpus. This means that the feature acquisition task do not require human intervention or expert knowledge. We will develop both methods of feature acquisition and the notion of n-grams and generalized vs-grams respectively in Sections 2.4.1 and 2.4.2.

Once all the features determined by one of these methods (automatic or semi-automatic), we represent the lines of the corpus by a binary vector in the vectorial space obtained by the set of features. For a given vector, the value of each element corresponds to the presence or absence of a feature. This representation allows us to build a training dataset based on the features. We use this dataset in a supervised classification system to obtain the rules (models) for the recognition of logical structures. This enables to determine the different combinations of features in order to predict the logical divisions. The trained classification model is subsequently used in order to associate the lines of a new non-annotated corpus with a class: Class of lines representing a logical division (*positive*), and class of lines which are not associated with a logical division (*negative*).

2.4 Representation of logical divisions of log files

In this section, we present the steps to characterize and find the complex logical divisions: the feature acquisition and the classification model creation. We also introduce the original notion of generalized vs-grams, used in the automatic acquisition of features, which are the main contribution of our segmentation approach.

2.4.1 Heuristics Based Feature Acquisition

First, we assume that the identification of the beginning of a segment is sufficient to recognize a logical unit in a document. The end of a logical unit is determined by the start of the next one. Thus, we seek to identify *syntactic patterns* which exist in lines representing the beginning of a logical unit. These syntactic patterns allow to *differentiate* the beginning lines of logical unit (i.e., logical divisions) from other lines.

```
1
    set_ top i:/WORK/fifo
Setting top design to 'i:/WORK/fifo'
Status: Implementing inferred operators
```

Figure 2.1: A logical unit in a log file.

Figure 2.1 shows an example of a logical unit in the log file log_B (see Figure 1.5 on page 16). The highlighted line represents the beginning of the unit. In a simple way, we can characterize the beginning of this logical unit by a pattern as "`<abs-shift> <string>< :><string>`"[2] which means a line beginning with an absolute shift (indentation), followed by a string, then a ":", and finally followed by another string. We consider the *couple* of this pattern and its position around the beginning of the segment as a feature whose the presence helps to recognize this logical unit.

However, according to the domain characteristics and especially the heterogeneity that exists in such documents, we must characterize the beginning of segments more accurately. This means that it is necessary to characterize a logical unit by using a sufficient number of relevant features. To build this set of features, we identify syntactic patterns in a window of lines around the beginning of logical units. The window size being a parameter of our approach: "*nlp*" represents the number of lines before the beginning of the segment, and "*nls*" is the number of lines following the beginning of the segment. In the case of log files studied here, the values of these two parameters are fixed following an experimental protocol. The used values are presented in Section

2. abs-shift: Absolute shift (indentation) at beginning of line.

2.6.

We detail below the constitution of features based on an example. Figure 2.2 shows another fragment of the log file log_A (see Figure 1.4 on page 15). The highlighted lines (lines 86 and 92) represent the beginning of two logical units (segments).

```
84 | Total IO Pad Cell Area                : 1076208.64
85 |
86 | ------ Design Statistics:
87 |
88 | Number of Instances                         : 13628
89 | Number of Nets                              : 14293
90 | Maximum number of Pins in Net                 : 531
91 |
92 |  IO Port summary
93 | Number of Primary I/O Ports                   : 388
94 | Number of Input Ports                         : 259
```

Figure 2.2: Two logical units in the log file log_A.

To simplify the example, we use a window size of five lines around the beginning of each segment (i.e., *nlp*=2 and *nls*=2). Regarding the first segment, we can identify the pattern "`<--->``<string>``<fin :>`" on the beginning line of the segment (line 86). We also have the pattern "`<string> <:> <string>`" on the both second lines before and after the beginning of the segment (i.e., lines 84 and 88). We also consider the empty line as a pattern. Thus, considering the identified patterns and their locations, we obtain the following features for the first segment:

f_a(`<---><string><fin :>`, 0)
f_b(`<emptyLine>`, -1)
f_c(`<emptyLine>`, +1)
f_d(`<string><:><string>`, -2)
f_e(`<string><:><string>`, +2)

For each feature, the number after the pattern is the line number in the window. The zero corresponds to the beginning line of the segment. As an example, the f_a presents a feature consisting of the shown pattern, found on the beginning line of the segment (i.e., line 0 in the window around the beginning of the segment) which corresponds to the line 86 in the example. f_d also presents another feature consisting of the pattern found on second line before the beginning of the segment.

Regarding the second segment in Figure 2.2, we can identify some of the features found around the first segment. For instance, on the second line before the beginning of the second segment (line 90), we found the pattern "`<string><:><string>`" which has been also found on the same position (second line before the beginning of the segment) around the first segment. Thus, we also note the same feature (i.e., f_d) for the second segment. Meanwhile, we also identify two new features (i.e., f_f and f_g) identified on lines 92 and 93. We note the below features around the beginning of the second segment.

f_f(`<abs-shift><string>`, 0) f_d(`<string><:><string>`, -2)
f_b(`<emptyLine>`, -1) f_e(`<string><:><string>`, +2)
f_g(`<string><:><string>`, +1)

Finally, by combining all the identified features (for both segments), we obtain a set of features. The following list represents this set of features obtained on the example of Figure 2.2 = $\{f_a, f_b, f_c, f_d, f_e, f_f, f_g\}$

Then, by considering the feature set as a vector space, we represent both logical units in form of binary vectors. Therefore, the beginning of each logical unit of the example is represented by a binary vector as follows. Here, "1" means the presence of the feature and, "0" its absence.

$S_1 : \{1, 1, 1, 1, 1, 0, 0\}$ $S_1 : \{0, 1, 0, 1, 1, 1, 1\}$

To obtain the set of features, we established a corpus of different log files. The constitution of this corpus is done in collaboration with a domain expert to ensure that all domain logical units exist in the corpus. The determined feature set contains 123 features in total. Once the feature set constituted, we characterize each line of the documents in the vector space defined by the feature set. Since the logical units in the log corpus are annotated by an expert, we obtain the vectors of positive instances (the lines corresponding to the beginning of logical units) and negative instances (the lines that do not match the beginning of logical units). Then, we use this positive and negative instances in a supervised learning system to obtain a model for the recognition of logical units (see Section 2.5).

2.4.2 Features Acquisition Using the Generalized vs-grams

In the previous section, we have described a solution to build the feature set using syntactic patterns. Nevertheless, this method (see Section 2.4.1) requires expert knowledge to provide some heuristics and then define the patterns specific to them.

In the remainder of this work, we propose an automatic method that we use to create the set of features, without requiring human intervention. Thus, to create the set of features, we decided to use the n-grams to characterize the beginning of logical units in the log files. An n-gram is a set of *n* items in a sequence. In the field of NLP, an n-gram is a series of *n* items in a text where the items can be letters or words. N-grams are often used as features in the textual document classification tasks [Tan et al., 2002] to model the content and the sequence of words in a document.

But in our context, we are only interested in the structure of documents. That means we do *not* seek to identify the logical units according to their content (words or letters), *but* according to their textual structures (punctuation, symbols, layouts, etc.). This necessity led us to define and propose an *original kind of grams* that we call *generalized vs-grams*. Generalized vs-grams allow to model the textual structures (the layouts and the composition of letters and special characters) of a document while being insensitive to the contents of the latter. This enables to characterize the visual structure of a text document.

Generalized Vs-grams.

We present here the concept of *generalized vs-grams* that we defined in the context of this work. We choose the term *vs-gram* as an abbreviation for "*Variable Size Grams*". To better understand the concept of vs-grams, we first explain, via an example, the needs that led us to define vs-grams.

Figure 2.3 shows an extract from log file log_B. The highlighted lines (52 and 62) represent the beginning of two logical units.

```
52 | ****** Signal Coverages ******
53 | CV_ INT[0:255]                    No No
54 | ALG_ DATA_ INT [0:127]                No No
55 | END_ KEY                          No No
...|
62 | **** MODULE INSTANCE ****
63 | MODULE TB_ ECB_ VK_ DEC_ ITER.TOP_ LEVEL.CTRL
64 | FILE /users/AES/src/controller_ iter.vhdl
```

Figure 2.3: logical unit in a log file.

In this example, the beginning of the two logical units is characterized by a string preceded and followed by a series of the symbol "*". By extracting the fixed size n-grams (e.g., n = 3), we obtain the following tri-grams on the first segment[3]:

"***", "␣si", "gna", "l␣c", "ove", "rag", "es␣".

Similarly, we have the following tri-grams for the second segment.

"***", "*␣m", "odu", "le␣", "ins", "tan", "ce␣".

The extracted tri-grams show "the sequence of letters" in those two lines. However, the particular composition of special characters and letters which characterizes here the visual layout of these two logical units is hardly highlighted by the tri-grams. Indeed, the fixed size n-grams are not relevant when one focuses at layout and the composition of letters, symbols, and punctuation. In order to describe the layout and the composition of letters and symbols in a line by means of a kind of feature, we define vs-grams where grams can have variable size.

We aim at defining a kind of feature which characterizes how letters, whitespaces, and symbols are located in a text line from the point of view of their positions. This

3. the extracted grams are case-insensitive, as the letter case is not informative in this context.

makes it enable to describe the layout or visual structure of a line. Thus, we define vs-gram as a series of alphanumeric and non-alphanumeric characters whose boundaries are determined according to the type of seen characters. In order to determine the boundaries of a vs-gram, we define three conditions which make it enable to describe how the alphanumeric and non-alphanumeric characters follow each other in a line (i.e., visual structure of a line). Therefore, a vs-grams is a series of alphanumeric and non-alphanumeric characters, which is defined as follows:

– If the gram contains a series of alphanumeric characters, it ends with a non-alphanumeric character. The next gram begins with the non-alphanumeric character.
– If the gram starts with a series of non-alphanumeric characters, it ends with an alphanumeric character. The next gram begins with the alphanumeric character.

– if the seen character is a whitespace, it is systematically added to the current gram.
Taking the previous example (Figure 2.3), we obtain the following grams for the first segment:

`"******␣s"`, `"signal␣coverage␣*"`, `"******"`

Figure 1 presents the algorithm to extract the vs-grams. As shown in the algorithm, the extraction process consists in verification of three conditions: if the current character is a alphanumeric one, if it is a white space or finally it is non-alphanumeric character. Based on the type of the current char and that of previously seen char as well as the following one, we extract the vs-grams.
Contrary the previously extracted tri-grams, the vs-grams show the composition model of the letters and special characters (here, "*") and their positions according to each other in this line. That is, these vs-grams express a composition of characters as a string of letters surrounded by symbols "*". This *pattern identified by vs-grams* marks the beginning of a logical unit in the log files.

The vs-grams are still sensitive to the content of texts. For example, here, the second extracted vs-gram presents a series of letters *comprising* the words "signal"

Algorithm 1: Extraction of vs-grams in a text file

```
Data: $Text(T) = \{ch_0, ch_1, ch_2, \ldots, ch_n \mid n=|T| \text{ and } ch_i \in \{ASCI\ chars\}$
Result: LIST(vsgrams) : list of extracted vs-grams

for $i \leftarrow 0$ to $n$ do
    currentChar = $char_i \in T$;
    if currentChar is LetterOrDigit then
        if last char was not LetterOrDigit then
            Call AddGram();
        else
            gram ⟵ gram + currentChar;
    else if currentChar is Whitespace then
        gram ⟵ gram + currentChar;
    else if currentChar is SpecialChar then
        if last char was not LetterOrDigit then
            gram ⟵ gram + currentChar;
        else
            Call AddGram();

return LIST(vsgrams);
AddGram(){ gram ⟵ gram + currentChar;
LIST(vsgrams) ⟵ gram;
gram ⟵ currentChar }
```

and "`coverage`". However, the *essential knowledge* to take into account is the *presence of a string*. Similarly, the number of stars ("*") in the two other vs-grams, for example, is not informative. That is why we generalize vs-grams by replacing sequences of letters and special characters by some symbols representing their character type and a counter notion like "+"[4]. This means that we replace, for example, a string of alphanumeric characters with the symbol "`\w+`". The series of stars will be replaced by "`\*+`". Thus, in this example, we obtain the following generalized vs-grams: `"\*+␣s", "\w+␣\*+", "\*+"`

We finally obtain generalized vs-grams that express well the existence of a string of alphanumeric characters surrounded by a sequence of the "$*$" symbol.

4. "+" means one repetition or more.

The generalized vs-grams allow to generalize the content of the lines. This constitutes essential information while reducing the representation space compared to n-grams.

Once the notion of generalized vs-grams defined, we build the set of features by extracting the generalized vs-grams in a line window around the beginning of logical units. Like the previous solution, the generalized vs-grams are associated with line numbers (in the window) wherein they are extracted. A feature is hence a composition of a vs-gram and the position of the line (in the window) wherein it is extracted. The obtained feature set for the corpus of log files contains 1023 elements. The creation of positive and negative instances is conducted by using the feature set obtained by extracting generalized vs-grams in the tagged corpus of log files.

To exemplify the creation of vs-gram features, we use the same log file segments presented in Figure 2.2, which were previously used to extract manual features. Taking the first segment of Figure 2.2, we obtain the following features by extracting the generalized vs-grams:

f_1(\-+␣\w+, 0)	f_2(\w+ :, 0)
f_3(\w+ \s+ :, -2)	f_4(: \w+, -2)
f_5(\w+ \s+ :, +2)	f_6(: \w+, +2)

Feature f_1 contains a seen vs-gram (\-+␣\w+) on the beginning of the first segment. This vs-gram corresponds to "------ D" in the beginning of the segment (line 86). Then the rest of line is characterized by the next extracted vs-gram (\w+ :) which is used in feature f_2. Thus, the beginning line of the first segment, is characterized by the two features f_1 and f_2. Then, we have f_3 and f_4 which are composed of vs-grams extracted in the second line (line 84) before the beginning of the first segment. f_5 and f_6 correspond also to the second line after the beginning of the first segment.

2.5 Learning to Identify Logical Units

The objective of our work is to determine a model of rules from which we can identify the logical units in the log files. For this purpose, we first establish a corpus of log files. Then, we identify the set of features using the methods described in Sections 2.4.1 and

2.4.2. Each line of the corpus is seen as a positive or negative instance. The lines are represented as a boolean vector whose elements are features.

Afterwards, a supervised machine learning process based on a classification method is applied. The instances previously obtained are used as training set. The obtained classification model enables to classify the lines of corpus into two classes (beginning of segment / not beginning of segment).

Regarding the training data, we are faced with the balance of positive and negative instances problem. Indeed, in our corpus obtained from real data, the number of negative examples is seven times more than the number of positive examples. The distribution of "+" and "-" examples can influence a classifier that uses the probability of each class to predict. Moreover, the lack of examples for special cases prevents the classifier to create the necessary rules. To address this unbalance problem several solutions can be applied:

- "Over sampling"
- "Under sampling"
- "SMOTE"

In the "Over sample", examples of the class with fewer examples are duplicated. If we want to retain the same size for the new dataset, it requires removing some of the examples of the other class. The choice of examples to be duplicated or removed is performed randomly. Nevertheless a real situation may create a problem of over-fitting.

The solution "Under Sampling" consisting of removing some examples of a class with more examples, has its own drawbacks. Indeed, we cannot guarantee the conservation of all instances of a particular case in that class. Moreover, this does not solve the lack of examples in the other class.

The "SMOTE" method consists in generating synthetic examples. This generation can build synthetic positive examples that cannot exist in real case according to the characteristics of the domain. Indeed, we observed that in some cases, by changing the value of a single feature in a positive example, we obtain an example which is inconsistent in the real world.

Depending on the characteristics of our data, we finally chose the "over sampling" solution. The parameters of the algorithm like the size of the new dataset and the new class distribution rate are set after several tests. We tested several classification methods including "K Nearest Neighbors" (KNN) and "decision trees". The method of K Nearest Neighbors predicted the class of an element by taking into account the class of K training samples which are closest to the element, based on a defined distance. The decision tree is used to divide a population of individuals into classes, according to a set of discriminating variables. The classification model obtained in the learning phase will be used to identify the lines representing the beginning of logical units in real industrial data. The classification results and obtained models are developed in Section 2.6. We also discuss the impact of both methods of feature acquisition in the next section.

2.6 Experiments

The experiments are conducted in two main directions:

- performance of the classification of the log file lines into positive and negative classes:
  - using features built based on defined patterns
  - using generalized vs-gram features
- comparison of our segmentation method with the well-know TextTiling segmentation method.

We use the same training corpus in all experiments. The learning corpus consists in 19 different log files from the industrial world. The log files contain real data and differ in content and especially in structure. The training corpus size is 1.1 MB and contains in total 19,638 lines.

We have calculated the performance of our approach in terms of precision and recall of the classification model. Precision for a class is the number of true positives divided by the number of instances *predicted* as positive. Recall in a class is the number of true positives divided by the number of instances *actually belonging* to the class. Finally, the harmonic average between precision and recall is calculated by the F-Score.

We thus test the classification models obtained with each feature acquisition methods proposed in this chapter. As a reminder, in the classification, a line is a positive instance if and if it presents the beginning of a logical unit (i.e., logical division). A negative instance is a line which does not represent a logical division.

2.6.1 Tests Using Features Build by the Defined Patterns

At this stage of experiments, the feature set is created by the method described in Section 2.4.1. By using the training corpus to build the feature set, we identified 128 patterns. Consequently, we have 128 features to create instances. In the training data, there are 1,142 positive examples against 18,496 negative examples.

We present here the results obtained using the classification algorithms that yielded the best results: C4.5 decision tree [Quinlan, 1993] and KNN [Mitchell, 1997]. Although the results obtained by other algorithms like SVM are largely close to those obtained by KNN or C4.5, we do not provide the results of all tested algorithm as our objective if not to compare the performance of classification algorithm, but to evaluate the quality of build feature sets. In other words, we look to know if the extracted features (manually and automatically) are enough relevant to represent the visual characteristics of logical units. Their relevancy can be determined by the accuracy of the classification using these features.

To apply the classification algorithms, we use the built-in implementations in WEKA software[5]. To evaluate the classification performance, we use the cross validation (10 folds).

Cross-validation is a method for estimating reliability of a model based on a sampling technique. We divide the initial training set into “n" samples and repeat the learning “n" times while choosing n-1 samples as training set and the last sample for evaluation. Each sample is used once for evaluation. Finally, the performance is given by the average of performances obtained in each iteration of the process.

5. http://www.cs.waikato.ac.nz/ml/weka/

We firstly perform learning by using the “unbalanced" dataset and then by using a dataset balanced via the “over sampling" (cf. section2.5). Table 2.1 presents precision, recall, and F-score obtained for each class.

Class	Precision	Recall	F-Score
Pos	0,92	0,84	**0,88**
Neg	0,99	0,99	**0,99**

Class	Precision	Recall	F-Score
Pos	0,92	0,88	**0,90**
Neg	0,99	0,99	**0,99**

Table 2.1: Classification performance according to each class - C4.5 (left) and KNN (right) - using the *unbalanced* dataset - Features obtained by “defined patterns"

We point out that by using the KNN and C4.5 classifiers when the training set is created using the features built via “defined patterns", we obtain an F-Score between 0.88 and 0.99. We observe that the classification model learned from the unbalanced dataset is less accurate in classifying positive examples. This is because there are significantly fewer positive examples in the dataset than negative examples. To improve the classification performance, we hence balance the dataset.

In the following, we use the balanced dataset. For this, we applied the “over sampling" method of Weka. We chose 0.8 as the equilibrium rate of the number of examples. We finally got a dataset in which there are 5,833 positive examples against 9,877 negative examples. Table 2.2 presents precision, recall, and F-score obtained for each class.

Class	Precision	Recall	F-Score
Pos	0.98	0.99	**0.99**
Neg	0.99	0.99	**0.99**

Class	Precision	Recall	F-Score
Pos	0.98	0.99	**0.99**
Neg	0.99	0.99	**0.99**

Table 2.2: Classification performance according to each class - C4.5 (left) and KNN (right) - using the *balanced* dataset - Features obtained by “defined patterns"

We observe that, by balancing the dataset, we improved the precision and recall of positive examples. Moreover, the results show that with the features obtained by the semi-automatic method, we succeed in creating a set of rules (a classification model) with an overall F-score of 0.99.

2.6.2 Tests Using Generalized vs-grams Features

We experiment the use of generalized vs-grams as features (automatic method). For this, we use the same corpus described above. The tests are also conducted using unbalanced and balanced datasets.

Table 2.3 shows precision, recall, and F-Score obtained by each classifier using the unbalanced dataset where the features are obtained via the extraction of generalized vs-grams. Table 2.4 also shows the performance of classifiers using the dataset that we have balanced using the same parameters explained in Section 2.6.1.

Class	Precision	Recall	F-Score
Pos	0.87	0.64	**0.74**
Neg	0.97	0.97	**0.97**

Class	Precision	Recall	F-Score
Pos	0.87	0.67	**0.75**
Neg	0.97	0.98	**0.97**

Table 2.3: Classification performance according to each class - C4.5 (left) and KNN (right) - using the *unbalanced* dataset - Features obtained by "generalized vs-grams"

Class	Precision	Recall	F-Score
Pos	0.92	0.74	**0.82**
Neg	0.96	0.98	**0.97**

Class	Precision	Recall	F-Score
Pos	0.94	0.75	**0.84**
Neg	0.97	0.98	**0.97**

Table 2.4: Classification performance according to each class - C4.5 (left) and KNN (right) - using the *balanced* dataset - Features obtained by "generalized vs-grams"

The results show that balancing the dataset improves the classification performance and therefore the precision and recall in the recognition of logical units. For example with KNN, we obtain a precision equal to 0.94 in the positive class and equal to 0.97 in the negative class.

We find that the automatic method based on *generalized vs-grams* gives a classification performance close to the semi-automatic method. This means that the generalized vs-grams represents well the syntactic characteristics of logical units of a document. Finally, in order to improve the performance of classification, we have decided to create a set of features comprised mainly the automatic features (the generalized vs-grams), and a limited number of features from the heuristics of an expert (defined patterns). We have, in these experiments, ten features obtained by the heuristics. They characterize mostly the *first* and the *end* of the line representing a logical division. Table 2.5 shows the performance of classification and therefore the recognition of logical units using a new dataset obtained by this new set of mixed features.

Class	Precision	Recall	F-Score
Pos	0.98	0.99	**0.99**
Neg	1	0.99	**1**

Class	Precision	Recall	F-Score
Pos	0.98	0.99	**0.99**
Neg	1	0.99	**1**

Table 2.5: Classification performance according to each class - C4.5 (left) and KNN (right) - using the dataset obtained by set of mixed features.

The results show that we obtain a performance similar to that obtained when we used the built features based on the heuristics of an expert. This shows that a minimum addition of expert knowledge (only ten semi-automatic features) significantly improves the results.

2.7 Discussing Semantic and Discourse Segmentation

Here we aim at studying how a semantic segmentation method would behave in the context of log files. For this purpose, we use the well-known TextTiling semantic segmentation method. This method, presented in Section 2.2, is a topic based segmentation method. It tries to recognises the topic changes in documents based on term co-occurrences and lexical cohesion measures. In order to perform the tests, we use the David James implementation of TextTiling, which is used in Lingua NLP package[6].

What is a relevant segment?

Regarding the segments obtained by TextTiling, we observed that they hardly correspond to the segments initially tagged by a domain expert in the corpus of log files. In fact, we should consider that each segmentation method split a document into segments

6. http://search.cpan.org/~splice/Lingua-EN-Segmenter-0.1/lib/Lingua/EN/Segmenter/TextTiling.pm

which may differ from segments obtained by another method. This point is emphasized when the segmentation methods use different strategies and assumptions. In the same time, we note that it is possible to have different kinds of relevant segmentation for a document. This means that we cannot say that there is only one relevant segmentation model for a document.

In our context, the log files are initially tagged by an expert based on their structure whereas TextTiling determines segments based on the identified topic changes. That is why it is not relevant to compare the segments obtained by TextTiling with reference segments which are based on the log file logical structure. Therefore, we asked the domain expert again to independently analyze the relevance of segments obtained by TextTiling. In this analysis, a segment is considered *irrelevant* if:

– it contains several text chunks that each one should be considered as a unique segment;
– it does not contain the entire of a text chunk that represents an unique segment.

In the first case, the obtained segment should be split into more segments as it contains different text chunks which are not significantly correlated. The fact that there are different kinds of information in a segment can bias the passage retrieval performance. A relevant segment should only contain correlated information.

In the second case, the issue is more important as the obtained segment do not contain the totality of the information. This means that the segmentation method has split a relevant segment into two or more irrelevant segments. In such situation, we lost some part of information which is not situated in the obtained segment.

Quality of TextTiling segments

For segments obtained by TextTiling, we evaluate them based on the two previously presented conditions. Once the relevant and irrelevant segments are determined, we define precision as the number of relevant segments divided by the number of all obtained segments. Since in the case of TextTiling, there are not a reference segmentation model, we do not provide a recall estimation. In other words, we just seek to evaluate the relevance of segments obtained by TextTiling.

In the case of our proposed approach, by using the automatic method (extraction of generalised vs-gram) to acquire the features, we do not need the expert knowledge nor to adapt our approach to different kind of log files. Whereas TextTiling needs to be parametrized according to the characteristics of documents. The main TextTiling parameters are Pseudo-sentence size and size (in sentences) of the block used in the block comparison method. After some tests on log file corpus, we set these two parameters to "8" which determine the size of Pseudo-sentence in terms of word and the size of text blocks in terms of sentences. In order to note also the performance of our approach, we provide the results obtained by our approach. In our approach, we used the vs-gram features and KNN classification.

	Obtained segments	Irrelevant	Relevant	Precision
Segmentation using vs-grams	1086	56	1030	**94**%
TextTiling	377	145	232	**61**%

Table 2.6: Segmentation by structure recognition using generalized vs-gram vs. TextTiling method

As shown in Table 2.6, we observe that only 61% of segments obtained by TextTiling are relevant. Regarding our method, the obtained results show that 94% of determined segments are relevant.

By analysing the obtained segments, we observed that TextTiling is not capable to recognize all topic changes, which results in large segments which should be split into more small segments. In the same time, TextTiling did not recognize the entire of tables as an unique segments. Most tables are split into several segments. This situation is explained via an example in Section 2.2 while we discussed the relevance of existing topic based methods. Moreover, there is a visual structure in log files which helps to recognize the correlated information whereas the topic based methods, like TextTiling, which use the lexical cohesion measures do not take this visual structure into account.

2.8 Discussion

. In this chapter, we have presented an approach to segment the textual data like log files which consist of complex textual structures into relevant text chunks. A relevant text chunk, called segment, only includes correlated information.

In Section 2.1, we have first presented the notion of text segmentation and its application in other domains like information retrieval and question answering. Considering our main objective, i.e., locating information in log files, we demonstrated that we first require to segment log files into relevant text chunks. Since each segment addresses a single topic, we can efficiently locate a requested information in log files by analysing the content of each segment.

In this section, we also discussed why a segmentation independent of queries is more relevant in our case. Actually, in our context, a user can look for more than one hundred information in a corpus of log files in each execution, whereas the size of a given log file can simply reaches more than 1GB. In addition, the structure of log files is a relevant guide line to determine concise text segments. Thus, a query independent segmentation in the context of log files is first feasible and second we obtain a considerable performance gain as we do not require to perform segmentation for each query.

In Section 2.2, we have studied the existing work in the domain of text segmentation. We have introduced the three main directions, i.e., semantic segmentation, window-based segmentation, discourse passages. We argued why the methods based on semantic segmentation and window-based are irrelevant in the context of log files. We also discussed that the subject change based on the lexical cohesion assumption is irrelevant in this context. By means of few examples, we demonstrated that a change of co-occurrences or a change of term repetitions does not necessarily result in topic change in log files. This throws doubt on the relevance of assumptions used in semantic segmentation. We also supplied other reasons such as the presence of a considerable amount of numerical data and the lack of semantic resources in this domain. Then, we showed that a window-based segmentation can result in the lost of information or a potential cutting of a relevant chunk of a document into several segments. Also, it ignores the structure of log files although log file structures contain relevant information to identify segments. We finally presented the segmentation methods based on the discourse passages. The idea is to identify the logical units of documents as each logical units should address a single topic. As mentioned in [Llopis et al., 2002a], the discourse-based methods look more effective since they are using the structure of the document itself. Similarly, Callan notes in [Callan, 1994] that discourse passages are expected to be the most effective, because discourse boundaries organize material by content.

Nevertheless, we have also noted the drawbacks of discourse-based methods. Among these issues, we first noted that these methods need the documents to be written entirely in an consistent way and following a discourse logic. Second, the logical unit markers (i.e., logical divisions) usually are ambiguous with other types of separators. For example, *headers*, *list elements* or *table rows* might be separated in the same way as discourse related paragraphs (for example using an *empty line*). We also noted that there is some problems in dealing with special structures such as headers, lists, and tables which are easily mixed with other units such as proper paragraphs [Tiedemann and Mur, 2008].

In log files, there are structures such as tables, data blocks, and specific strings marking the beginning of new information. Moreover, the structural organization of log files is based on a defined grammar which guarantees a consistency in writing log files. Therefore, the first mentioned drawbacks of discourse segmentation has no place in the context of log file. However, the second problem is emphasised in log files where conventional logical divisions are meaningless (e.g., paragraph indentions) or ambiguous (e.g., empty lines). In addition, log files contain a considerable number of tables, lists, and data blocks.

Taking these points into account, our objective was to propose a segmentation solution based on the logical structure of log files which does not have the mentioned drawbacks of existing discourse-based methods. The main challenge was to take the ambiguous logical divisions and complex structures likes tables into account.

In Section 2.3, we presented our global process of identification and subsequently recognition of logical units. We subsequently detailed our approach in Sections 2.4 and 2.5. Our approach to identify logical units needs to recognize different layouts and data presentation formats used in log files. We hence looked for syntactic cues which indicate a separation of consecutive text chunks. Therefore, we characterized logical units according to their *syntactic characteristics*. We proposed two approaches (semi-automatic and automatic) to obtain features representing the syntactic characteristics of logical units. In the semi-automatic way, we first defined the features according to some heuristics based on an expert knowledge. In the case of the automatic method, we proposed an original type of n-grams, called "*generalized vs-grams*", which make it enable to characterize the layout and the composition of letters, symbols, and punctuation in textual documents. We defined vs-gram as a series of alphanumeric and non-alphanumeric characters whose boundaries are determined according to the "type" of seen characters.

In Section 2.6, we presented the test performed to evaluate each method of feature acquisition and finally the performance of segmentation. The results showed that by means of our approach we can identify different kinds of logical units in log files. The extracted vs-grams in log files make it enable to model the visual structure (layout) of logical divisions and thus to recognize them. Thanks to our solution we have overcame the drawbacks previously explained.

We finally performed in Section 2.7 some tests to evaluate the performance of semantic-based methods. For this purpose, we used the TextTiling method presented in Section 2.2. Analysing the recognized segments shows that TextTiling does not recognize all topic changes, which results in large segments which should be split into more small segments. Moreover, most tables are split into several segments whereas our approach recognized all tables.

After segmenting log files, we require to analyse their relevance and similarity to a given query in order to locate the requested information in log files. For this purpose, we will investigate a study on passage retrieval methods and the difficulties existing in the application of passage retrieval in the context of log files. We devote the next Chapter to this study, i.e., locating information in log files by means of passage retrieval.

Chapter

3

Passage Retrieval in Log Files

Research is to see what everybody else has seen, and to think what nobody else has thought.

Albert Szent-Gyorgyi

Preamble

This chapter is devoted to our work on Passage Retrieval in log files. We present how to enhance passage retrieval performance in this domain by expanding initial queries. We also present a novel term weighting measure which aims at assigning a weight to terms according to their relatedness to queries. This measure, called TRQ (Term Relatedness to Query), is used to identify the most relevant expansion terms. We also apply our approach to documents from general domains. Our work on passage retrieval and query expansion in the context of log files, introduced in this chapter, is published in [Saneifar et al., 2010b] and [Saneifar et al., 2010a].

Contents

3.1 Introduction

Passage Retrieval, representing an important phase in question answering, is the task of searching for passages which may contain the answer for a given question. As an accurate and reliable definition, a passage is a fixed-length sequence of words which can begin and end anywhere in a document [Ofoghi et al., 2006]. Passage retrieval has been a research subject since 1970. However it has been extensively investigated since late 1980 and early 1990's [O'Connor, 1975], [Wade and Allan, 2005].

Text retrieval methods are typically designed to identify whole documents that are relevant to a query. In these approaches a query is evaluated by using the words and phrases in each document (i.e., the index terms) to compute the similarity between a document and a query [Kaszkiel and Zobel, 1997]. This means that by means of text retrieval methods, we are not able to locate the seeking information in documents, but to find only the relevant documents (i.e., documents containing the seeking information). As an alternative, we have passage retrieval which makes it possible to locate the requested information within a document. In this context, each document is seen as a set of passages, where a passage is a contiguous block of text. Thus, in order to retrieve passages containing the sought information, the similarity of each passage to a query is calculated. In fact, passage retrieval can be considered as an intermediate between document retrieval and information extraction. Therefore, the main objective is to locate sought information within documents and thus reduce the search space wherein we will look to extract the exact information. The passage retrieval phase significantly influences the performance of QA systems because final answers are sought in the retrieved passages. In other words, if a relevant passage is not retrieved, there will be no chance to answer the corresponding question.

The main differences between different passage retrieval systems are the way that they select the passages. That is to say, what they consider as a passage and the size of them [Tellex et al., 2003b].

In Chapter 2, we have discussed the different approaches to define and determine passages (text segments) in documents. We have subsequently presented an approach to split log files into text chunks (segments) based on the their logical structure. That provides us a collection of relevant segments of log files where each segment treats an amount of correlated information. The challenge is now to find log file segments that may contain the requested information. But the particularity of the studied data (i.e., log files) and characteristics of restricted domains significantly impact the accuracy and performance of passage retrieval in this context. We discuss in detail the impact of log file characteristics on the performance of passage retrieval in Section 3.2.

Due to the fact that log files are multi-source and multi-vocabulary data, the main challenge is the existing gap between vocabulary of queries and those of log files. We call this problem mismatch vocabularies. This issue is also noted in some other work. For example, the authors of [Buscaldi et al., 2010] note that the answer to a question may be unrelated to the terms used in the question itself, making classical term-based search methods useless [Buscaldi et al., 2010]. Because the user's formulation of the question is only one of the many possible ways to state the seeking information, there is often a discrepancy between the terminology used by the user and the terminology used in the document collection to describe the same concept [van der Plas and Tiedemann, 2008]. This issue is highlighted in the case of log files which are by default multi-vocabulary data.

Also, we have to deal with other challenges in passage retrieval from log files. We can briefly note the lack of data redundancy and thus lack of answer repetitions, lack of paraphrasing or surface patterns in log files, and the lack of semantic resources. We discuss and develop all these issues as well as mismatch vocabularies problem in Section 3.2.

Taking all these difficulties into account, we finally choose Query Expansion in order to improve the performance of passage retrieval in log files and overcome this domain problems notably mismatch vocabularies. Query expansion (or query enrichment[1]) attempts to improve retrieval performance by reformulating and adding new correlated terms to queries. In general the idea is to add more terms to an initial query in order to disambiguate the query and solve the possible term mismatch problem between the query and the relevant document [Keikha et al., 2011]. Here we present a query expansion

1. We use query expansion and query enrichment interchangeably in this paper.

approach using two novel relevance feedback levels.

The relevance feedback process, introduced in the mid-1960s is a controlled, automatic process for query reformulation, that can prove unusually effective [Salton and Buckley, 1997]. Relevance feedback is a powerful technique whereby a user can instruct an Information Retrieval system to find additional relevant documents by providing relevance information on certain documents or query terms [Selberg, 1997]. The basic idea behind relevance feedback is to take the results initially returned from a given query and to use information about whether or not those results are relevant to reformulate a new query.

Relevance feedback refers to an interactive process that helps to improve the retrieval performance. This means that when a user submits a query, an information retrieval system would return an initial set of result documents and then ask the user to judge whether some documents are relevant or not; after that, the system would reformulate the query based on the user's judgements, and return a set of new results. There are principally three types of relevance feedback: explicit, pseudo (blind), and implicit.

When there are no real relevance judgements available, alternatively, pseudo relevance feedback may be performed, which simply assumes that a small number of top-ranked documents in the initial retrieval results are relevant and then applies relevance feedback. Besides, somewhere in between relevance feedback and pseudo relevance feedback is implicit feedback, in which a user's actions in interacting with a system are used to infer the user's information need [Lv and Zhai, 2009]. All these methods will be detailed in Section 4.2.

Our query expansion based on relevance feedback involves two levels. In the first one, we implement an explicit relevance feedback system. The feedback is obtained from a training corpus within a *supervised learning approach*. We propose a new method for learning the *context* of questions (queries), based on the "*lexical world*" notion. Then, the contexts of questions are used as relevant documents wherein we look for expansion terms. Indeed, expansion terms are determined by identifying informative terms *representing* the *context* of questions.

The second phase consists in a novel kind of pseudo relevance feedback. Contrary to most pseudo relevance feedback methods considering the initial top-ranked documents as relevant, our method is based on a *new term weighting function*, called TRQ[2], which gives a score to terms of corpus according to their *relatedness* to the *query*. Indeed, we present the TRQ measure as an original term weighting function which aims at giving a high score to terms of the corpus which have a significant probability of existing in the relevant passages[3]. Terms having the highest TRQ scores are selected as expansion terms. This phase of query expansion is executed during the passage retrieval phase and by using the test corpus. That is why there are not any explicit information about the relevance of the passage at this stage.

We also evaluate the application of our approach in general domains. We thus use the documents used in TREC evaluation campaigns. We study the difference between the application of our approach in specific and general domains. We show that our approach gives *satisfactory* results on *real data* from the industrial field as well as general domains.

In Section 3.2, we present the main characteristics of log files which rise some challenges in passage retrieval. Existing work concerning passage retrieval systems and applications of relevance feedback in the query expansion are presented in Section 4.2. Section 3.4 presents some notions used in enrichment processes and also the first level of query enrichment. In Section 3.5, we develop our query expansion approach by presenting our novel term weighting function. Section 3.6 is devoted to developing the application of our approach in open domains. Experiments on real data are presented in Section 3.7. Finally, in Section 3.8, we present a discussion on the approach proposed in this chapter.

3.2 Difficulties in Passage Retrieval in log files

The particular characteristics of logs, described below, rise some challenges in Passage Retrieval and Information Extraction in log files. Here, by presenting these

2. Term Relatedness to Queries.
3. Passages containing answers to questions.

challenges and difficulties, we explain how they led us to query expansion as a solution.

First, we focus on problems arising from the multi-source aspect of log files. As previously explained, in the design of Integrated Circuits, different design tools can be used at the same time, while each tool generates its own log files. Therefore, although the logs of the same design level contain the *same* information, their *structure* and *vocabulary* can vary significantly *depending* on the *design tool used*. On the other words, each design tool has its own vocabulary to report the same information. This means that questions (queries) which are expressed using a vocabulary could not necessarily correspond to the vocabulary of all tools, or the query terms do not necessarily exist in the corresponding answers. We explain this issue, called here mismatch vocabularies, by the help of an example.

Consider the sentence "`Capture the total fixed STD cell`" as a given query and the two log files, log_A and log_B (see Figures 1.4 and 1.5 on pages 14 to 16), generated by two different tools, as the data resource wherein we look for answers.

The answer to the question, in log_A, is expressed in the third line of the following segment. This segment is situated in lines 103 to 107 in Figure 1.4.

```
103  Std cell utilization:  0.93% (449250/(48260400-0))
104  Chip area:             48260400 sites, bbox (210.00 210.00 4140.00 4139.60) um
105  Std cell area:         449250 sites, (non-fixed:449250 fixed:0)
106                         31625 cells, (non-fixed:12345 fixed:130)
107  Macro cell area:       0 sites
```

While the answer, in log_B, is expressed in the last line of the following segment which is located in lines 66 to 69 in Figure 1.5.

```
66  *** Clustered Preplaced Objects ***
67  # of preplaced io is: 18
68  # of preplaced hard macro is: 0
69  # of preplaced standard cell is: 24678
```

As shown above, the same information in two log files produced by two different tools is represented by different terms. The keywords of the question (i.e. `fixed`, `Std` & `cell`) exist in the answer extracted from log_A while the answer from log_B contains only the word "`cell`". Insofar as there is a *dictionary* associating the word "`Std`" with "`standard`", we can also take the word "`standard`" into account. However, if we make a query by using these two keywords, existing in the answer located in log_B, and send it to an IR system on log_B, irrelevant passages of log_B are retrieved.

For example, here we retrieve the following passage located in lines 57 to 60 in log_B wherein the keywords occur more frequently.

```
57 | *** Clustered FPlan Seeds ***
58 | # of standard cell seeds is: 0
59 | # of soft standard module seeds is: 19
60 | # of instance cell group seeds is: 8
```

This is because there is a significant *vocabulary mismatch* problem among log files log_A and log_B. This means that the query is expressed using the vocabulary of log_A which is different from the vocabulary of log_B. In other words, for a given question, *relevant answers* found in the logs of *some tools* do *not* necessarily contain the *keywords* of the question. Therefore, the initial query (*created by taking the keywords of the question*) may be relevant to logs generated by a tool (e.g., here log_A) but irrelevant to logs generated by another tool (e.g., here log_B) whereas we aim to answer questions regardless of the type of tools that generate the log files.

The *existence of question keywords* (or their syntactic variants) in a *passage* is an important factor to assess the *relevance* of the passage. Approaches based on the notion of common terms among questions and passages are detailed in Sect. 4.2. We note that the vocabulary changes in log files are not due to lexico-semantic variation of terms, but usually different instance/notion nomination or standardization. Moreover, there is not any relevant domain dictionary or semantic resource in the context of EDA in order to automatically identify corresponding terms.

Actually, the performance of a QA system depends largely on *redundant occurrences of answers* in the corpus in which answers are sought [Brill et al., 2001][Lin, 2007]. The methods developed for QA systems are generally based on the assumption that there are several instances of answers in the corpus. Tiedemann also argues in [Tiedemann, 2007] that *high redundancy* is desired in order to get as many relevant passages as possible to make it easier for the answer extraction modules to spot possible answers. He notes that high redundancy reduces the likelihood of selecting the wrong answer string by providing stronger evidence for the correct ones.

Regarding the importance of answer redundancy, an analysis of TREC9 experimental results in [Clarke et al., 2001] indicates that redundancy played an important role in the selection of answer fragments. Their experiments demonstrate that redundancy is an effective method for ranking candidate answers. In the same way, some evidence are provided in [Light et al., 2001a] that show the precision of a QA system depends largely on the redundancy of answer occurrences in the corpus.

Whereas information is rarely repeated in the log files of IC design tools. This means that for a question, there is only *one occurrence of the answer* in the corpus and thus one relevant passage containing the answer. Thus, methods based on exploiting the redundancy of answers in a corpus are not relevant. In such situation, a highly accurate passage retrieval module is required because the answer occurs in a very small set of passage [Ferrés and Rodríguez, 2006].

Taking all these evidences into account, the fact that there is only one occurrence of answer in log files, rises important challenge in locating the answer in log files. Our passage retrieval component has to achieve a high precision without being dependant on the answer redundancy.

Moreover, we note that if we do not retrieve the only existing relevant passage for a question among the top ranked candidate passages, we will not be able to extract answer at all. Whereas, when there are multiple relevant passages, we have more chance to retrieve at least one of the relevant passage among the top tanked candidates. Thus, it is necessary to find other solutions in order to enhance the performance of passage retrieval in the context of log files.

In addition, as previously said, design tools change over time, often unexpectedly. Therefore, the *data format* in the log files changes, which makes automatic data management difficult. Moreover, the language used in these logs is a difficulty that impacts information extraction methods. Although the language used in these logs is English, their contents do not usually comply with "*conventional*" grammar. In the processing of log files, we also deal with multi-format data: textual data, numerical data, alphanumerical, and structured data (e.g., tables and data blocks) as we have seen in Chapter 2.

Most QA systems for a given question, extract a large number of passages which likely contain the answer. But an important point in QA systems is to limit, as much as possible, the number of passages in which the final answer extraction is performed. If a passage retrieval module returns too many irrelevant passages, the answer extraction module is likely to fail to pinpoint the correct answer due to too much noise [Cui et al., 2005]. Since we are situated in a very specialized domain, high precision in the final answers (i.e. the percentage of correct answers) is a very important issue. This means that our passage retrieval system has to classify relevant passages (based on a relevance score) in the first positions among all retrieved candidate passages.

We therefore suggest expansion of initial queries in order to make them relevant to all types of logs (*generated by any kind of design tools*). By query expansion, we consider to enrich an initial query by integrating semantically similar terms coming from the vocabulary of different log files. That enables us to have a passage retrieval system which works well regardless of the type of supplied log files. This query expansion also improves the performance of passage retrieval in terms of precision as we obtain more relevant queries.

3.3 Related Work

In this section, we study the background work regarding passage retrieval and query expansion. We first introduce the main methods used in passage retrieval. Then, we focus on two main directions in query expansion. We finally provide a discussion about the relevance of cited methods in the context of log files.

Passage Retrieval and Locating Answers

Most passage retrieval algorithms depend on occurrences of query keywords in the corpus [Ofoghi et al., 2006]. We call this kind of methods statistic as they principally use metrics based on the occurrence numbers of query keywords in passages.

To find relevant passages, [Lamjiri et al., 2007] evaluate each passage using a scoring function based on the coverage of "question keywords" which also exist in the passage. In this category, we can note the overlap algorithm used in MITRE system [Light et al., 2001b]. MITRE is simply based on the number of terms that a passage has in common with the query. A sentence is here considered as a passage.

The term-density is also used in many work to retrieve the relevant passages. For example, in MultiText [Clarke et al., 2000] passage retrieval algorithm, which is a density-based one, retrieves short passages containing many terms with high IDF values[4]. In fact the relevancy of each passage is based on the number of query terms in the passage as well as the passage size which starts and ends with a query term [Tellex et al., 2003a].

In term-density based methods, the density is first determined by the extraction of question objects: lemmas of words, the types of named entities, and the type of answer to search. Then, for each element, an average distance is calculated between the current object and other objects of the question. The distance between terms is usually defined as the number of words occurring between them [Monz, 2003]. This distance is then used to calculate the density score to identify the passage which is related to the question [Gillard et al., 2006].

Other passage retrieval systems, such as those employed in SiteQ [Lee et al., 2001]

4. The IDF (Inverse Document Frequency) measure is explained in section 3.4.1.

and IBM [Ittycheriah and Roukos, 2002], are density-based as they take into account the distances between question terms in the candidate passages. SiteQ passage retrieval algorithm computes the score of an n-sentence passage by summing the weights of the individual sentences. Sentences are weighted based on query term density. This algorithm weights query terms based on their part of speech [Tellex et al., 2003a]. IBM's passage retrieval algorithm [Ittycheriah and Roukos, 2002] computes a series of distance measures for the passage notably "matching word measure", "mis-match word measure", and "cluster word measure". The first one computes the sums of IDF values of terms (or their synonyms) in the query whose appear in the passage. Mis-match word measure also functions in the same way but for query terms that do not appear in the passage. Finally, the "cluster word measure" counts the number of words that occur adjacently in both the question and the passage. These various measures are linearly combined to give the final score for a passage [Tellex et al., 2003a].

In the category of statistic methods, there are also methods which take different variations of keywords into account. For example, the use of morphological and semantic variants of query keywords is studied in [Chalendar et al., 2002]. [Kosseim and Yousefi, 2008] presents different approaches to take also semantic variations into account in order to complement the syntactic variants. Semantic variations are usually determined by using semantic resources like thesauri and based on the lexical relations like synonymy. In order to consider the lexical distance between questions and answers, [Moldovan et al., 2003] take the semantic and lexical variations of question keywords into account.

Question reformulation based on *surface patterns* is a standard method used to improve the performance of QA. The technique is based on identifying various ways of expressing an answer given a natural language question [Kosseim and Yousefi, 2008]. For example, for a question like "*Who founded the American Red Cross?*", QA systems based on surface patterns seek reformulations like "*the founder of the American Red Cross is X*" or "*X, the founder of the American Red Cross*". The question reformulation using surface patterns is also exploited in TREC9 and TREC10. Michael Kaisser also explores how lexical resources can be used to recognize a wide range of syntactic realizations that an answer sentence to a given question can have [Kaisser, 2009].

Query Expansion

QA systems also use query expansion methods to improve the retrieval performance. Delphine Bernhard argues in [Bernhard, 2010] that query expansion attempts to solve the vocabulary mismatch problem by adding new semantically related terms to the query. The query expansion is also studied among TREC participants. [Manning et al., 2008] presents two types of Query Expansion methods: (1) global and (2) local techniques. In the following paragraphs, we introduce some background work in both categories.

The global methods are based on external lexical-semantic resources such as ontologies. Most of QA systems use a kind of knowledge base in order to identify different lexical variations of question terms [Yang and Chua, 2002]. These variations are used in order to better analyse and retrieve relevant passages and extract answers. At this level, the systems typically involve morphological and semantic knowledge from existing electronic dictionaries and lexical resources such as WordNet[5]. Laurie, et. al. [Laurie et al., 2000] report improved retrieval quality using a form of controlled synonymy based on WordNet.

In fact, in general, there could be multiple words in the question and answer that are connected by many hidden causes. The causes themselves may have hidden causes associated with them. These causal relationships are represented in ontologies and WordNet [Paranjpe et al., 2004]. Deepa Paranjpe et al. propose in [Paranjpe et al., 2004] an aesthetically "clean" Bayesian inference scheme for exploiting lexical relations for passage-scoring for QA. They use the English WordNet to “bridge the gap” between query and response.

Magnini and Prevete describe in [Magnini and Prevete, 2000] an approach where they take the terms from the original question and add to the query morphological variants and synonyms of the original terms and the morphological variants. Pasca and Harabagiu [Pasca and Harabagiu, 2001] present an approach in which queries are expanded using morphology, lexical derivations and semantic equivalents, a kind of highly-controlled synonymy based on hand-maintained resources.

5. WordNet is a large lexical database of English. Nouns, verbs, adjectives, and adverbs are gathered into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. cf. http://wordnet.princeton.edu/

Agichtein et al. [Agichtein et al., 2001] propose a query expansion based on the Web resources. They take a number of keywords from the original question to form queries, which are expanded with phrases that are likely to occur in a declarative sentence that contains an answer. For instance, the question What is a binturong? is transformed into the queries: binturong 'refers to', binturong 'is a', and binturong 'is usually', which are then posted to a web search engine such as Altavista[6] or Google[7].

Relevance Feedback. The local methods are known as relevance feedback. As explained before, they use the results obtained for an original query in order to expand it. The application of relevance feedback depends largely on the information retrieval model. Most relevance feedback methods are based on the notion of Rocchio's ideal query [Rocchio, 1971]. The notion of Rocchio's ideal query can be described as a query that has maximum similarity to relevant documents and minimum similarity to irrelevant documents.

Xu and Croft introduce in [Xu and Croft, 2000] a relevance feedback (local) and global based query expansion method which is called local context analysis. That is, the selection of expansion terms is enhanced by considering concepts in the top-ranked documents that frequently co-occur with "many" query terms in the whole collection. Compared to classic relevance feedback, candidate expansion terms are more relevant to the query, as they have been observed to co-occur frequently in all documents [Wu et al., 2011]. Thus, expansion terms are selected based on their co-occurrences with query terms rather than based on their frequencies in the top-ranked documents.

Carpineto *et al* propose in [Carpineto et al., 2001] an other method that uses information theory measures for relevance feedback based query expansion. The main hypothesis of this method is that the difference between the distribution of terms in a set of relevant documents and the distribution of the same terms in the overall document collection reveals the semantic relatedness of those terms to the query.

In most relevance feedback systems, we first use the initial query to retrieve some

6. www.altavista.com
7. www.google.com

documents. Among the initially retrieved documents, those indicated by a user as relevant (explicit feedback) or only a few top-ranked ones (blind feedback) are analysed to extract expanding terms.

In an explicit system, feedback is obtained by some explicit evidence showing the relevance of a document. This type of feedback is explicit only when the assessors know that the feedback provided is interpreted as relevance judgements. This kind of relevance feedback can be considered as a kind of supervised machine learning.
For implicit feedback, there is not any direct information indicating the relevance of a document. The relevance of documents is inferred from user behaviours like the time spent on a document. This kind of feedback also needs the indirect user or expert intervention.
However, pseudo relevance feedback, also known as blind relevance feedback, provides a method for automatic local analysis. In fact, when there are no real relevance judgements available or user interaction cannot be studied, alternatively, pseudo relevance feedback may be performed. Usually, in blind relevance feedback, it is simply assumed that a small number of top-ranked documents in the initial retrieval results are relevant and finally the relevance feedback is based on this assumption [Lv and Zhai, 2009].

Existing query expansion techniques are very sensitive to the number of documents used for pseudo feedback [Li and Zhu, 2008]. Most approaches usually achieved the best performance when about 30 documents are used for pseudo feedback. Thus, Xiaoyan Li proposes in [Li, 2008] a robust relevance model based on a study of features that affected retrieval performance. These features included keywords from original queries, relevance ranks of documents from the first round retrieval, and common words in the background data collection.

Although most of question expansion methods are based on relevance feedback that is given automatically or manually at the moment of ad hoc retrieval, few techniques of query expansion use pre-categorized training documents to prepare expansion terms for domain specific search [Okabe and Yamada, 2007]. An approach based on the transductive learning is proposed in [Okabe and Yamada, 2007] in order to overcome the lack of relevant information. Transductive learning is a machine learning technique based on the transduction that creates classification labels for test data directly without making any approximate function from a training data [Vapnik, 1998]. This learning approach is more effective than traditional inductive learning, especially when there are few training examples. In this approach, after the user selects a relevant document, a transductive learning algorithm is used to find relevant documents the user did not find. It carries out the retrieval process by assigning a label (relevant or irrelevant) to each unjudged (unlabeled) document based on a small set of judged (labeled) data from the previous step.

Discussing the background methods

We have identified the main directions and methods in passage retrieval and query expansion according to the background work in these domains. These directions are presented in Table 3.1. In the following paragraphs, we discuss the relevance of these methods in the context of log files.

NLP & Statistic Methods. Despite the satisfactory results achieved by using surface patterns and syntactic variants in the mentioned work, these methods are irrelevant in the context of log files. Indeed, the main reasons for the irrelevancy of such methods are related to the fact that an answer is not reformulated in different ways in a corpus of log files. Moreover, there is a *lack of redundancy of answers* in log files. In addition, there are several technical and alphanumeric keywords in the domain of log files for which the use of syntactic or semantic variants appears to be complex or meaningless. Note that identification of lexico-semantic variations of terms usually requires an external knowledge base like ontologies or dictionaries which are not available in the context of studied log files.

Global Query Expansion based Methods. Regardless of the performance issue, in such methods, one has to tackle problems of semantic ambiguity, which explains why *local analysis* has been shown to be generally more effective than *global analysis* [Xu and Croft, 1996]. Moreover, as mentioned above, in order to determine the semantic relations

	NLP & Statistic methods				Global QE-based methods			Local QE-based methods	
	Term occurrence frequency	Term density	Morpho semantic variants	Surface patterns	Web	WordNet	Lexico semantic resources	Relevance Feedback	Machine learning
[Light et al., 2001b]	√								
[Clarke et al., 2000]		√							
[Gillard et al., 2006]		√							
[Lee et al., 2001]		√							
[Ittycheriah and Roukos, 2002]		√	√						
[Chalendar et al., 2002]	√		√						
[Kosseim and Yousefi, 2008]			√	√					
[Laurie et al., 2000]			√			√			
[Pasca and Harabagiu, 2001]			√				√		
[Paranjpe et al., 2004]						√			
[Magnini and Prevete, 2000]			√				√		
[Kaisser, 2009]			√	√			√		
[Agichtein et al., 2001]				√	√				
[Xu and Croft, 2000]	√							√	
[Carpineto et al., 2001]	√							√	
[Okabe and Yamada, 2007]								√	√
[Li, 2008]								√	

Table 3.1: Main directions in Passage Retrieval, Answer locating and Query Expansion to enhance the performance of QA systems

we need semantic resources like WordNet or a controlled vocabulary and categorised named entity list. In many specific domains, like in our case, there are no external lexical-semantic resources and their creation is a time-consuming task requiring domain expert knowledge. In addition, we could not supply a synonymity or lexical knowledge within our system and be sure that it covers all vocabularies may be used in this domain or by all types of log files. Thus, we decided to select expansion terms based on the on-going log files which are used and processed in the execution time in the system.

Local Query Expansion based Methods. In an explicit feedback case, in order to perform an initial retrieval, we need to index the training corpus, which can be time-consuming for industrial usage. In addition, we noticed that the (initially retrieved) passages are long enough to contain terms which are informative in the passages, *but* they are not really correlated to the query terms. That is why we propose to identify small fragments of log files which represent the context of queries to reduce the search space. We develop the query context identification in Section 3.4.

There are also some difficulties in using classic methods of blind relevance feedback. Indeed, in log files, we have observed that passages retrieved using the initial queries are not always relevant. This issue, i.e. the top-ranked initially retrieved documents are not always relevant, is also noted in other work like [Li and Zhu, 2008]. In our context, queries are actually questions predefined without caring about the type of data resources (log files) in which one looks for answers. That is why we could not consider the initially top-ranked passages as relevant when we do not have any external knowledge about their relevancy (blind case). That led us to define and propose a new way to select the expansion terms when the explicit relevance feedback does not exist. Our method which is based on a novel term scoring function, called TRQ measure, gives a high score to terms related to a query in a corpus. This new measure and our second phase of query enrichment is developed in Section 3.5.

As shown in [Xu and Croft, 2000], considering also the co-occurrence factor in local query expansion methods give better results. This feature is also reflected in our query expansion method by taking the dependency between terms and queries into account.

3.4 Passage Retrieval Enhancing by Query Enrichment

Our query enrichment approach is based on a *context learning* process and is associated with an *original term weighting function*. Our protocol of context learning is designed to determine the context of a given question by analysing the terms[8] co-occurring around the keywords of the question in the corpus. The new scoring function proposed here identifies terms related to the answers.
The architecture of our approach consists of three main modules:

1. Enrichment of the initial query by context learning (explicit feedback)
2. Enrichment by terms which are likely related to the answer (pseudo feedback)
3. Passage retrieval using the enriched queries

The first module is considered as a kind of *explicit* relevance feedback. The goal is to enrich initial queries extracted from questions in natural language so that they will be relevant to all types of log files generated by different tools. At this step, after *identifying the context of a question* within a supervised learning process, we enrich the initial query by the most significant terms extracted from the context.

The second module is designed for a second query expansion in order to obtain higher accuracy. At this phase, we aim at identifying terms which are likely related to the query in order to integrate them into the initial one. The objective is to make queries more flexible and relevant just before the Passage Retrieval process while there is significant difference and heterogeneity between the vocabulary of the training corpus and that of test corpus. Indeed, within the learning process, the enriched query is relevant to the most kinds of log files. However, in industrial applications, we can be faced with some cases where a second query expansion phase can improve the results on log files that are significantly different from those used in the training corpus. This can happen, for example, when a user supplies some log files as resources which differ considerably from those used for training. By means of the second enrichment, we avoid a new constitution of the training corpus and learning process when the system unexpectedly

8. In this chapter, the word "term" refers to both words and multi-word terms of the domain.

needs to process log files that are significantly different from those used in the training. Moreover, while expert knowledge is not available to build the training corpus and obtain explicit feedback, the second module can be used independently to expand the initial queries. Nevertheless, we point out that the second phase is not designed to replace the learning process (first enrichment phase).

In the third module, we seek the relevant passages in the logs generated by a tool that differs from that which has been used in the learning phase. That is, we have two different corpora of logs. The first one (called the *training corpus*) is used in the learning phase and the second one (called the *test corpus*) is used to retrieve relevant passages according to given questions. The logs of the test corpus have structures and a vocabulary that differ from the logs of the training corpus. We note that this part of our approach is different from machine learning based approaches. In fact, the training corpus is used to select the explicit relevance feedback from. This means that the training corpus is not a pre-annotated corpus. Moreover, the main idea in our approach is to learn the context of questions using a training corpus. The idea is not to learn rules based on which we can retrieve relevant passages in other corpus.

In our approach, we look for specialized context (hereafter called "*lexical world*") of question keywords. In order to characterize and present the specialized context of keywords, we use the terminological knowledge extracted from logs. Before explaining the query enrichment processes, we develop the concept of "lexical world" and the use of terminological knowledge in the following subsections.

Lexical World

The lexical world of a term is a small fragment of a document in which the term is seen. By determining the *lexical world* of a term in a document, we identify *terms* that *tend* to *appear around it* in that document. Actually, terms located around a term (within a small fragment) generally have strong semantic and/or contextual relations. We do not put any limit on the size of lexical worlds (eg., a few lines, a few words, etc.) as it has to be determined pragmatically based on the type of documents. Considering *n* words which surround a word as its context is also used in some proximity-based methods [van der Plas and Tiedemann, 2008]. However our work differs first in the way we characterize the context of keywords and second in how we finally determine the context of questions wherein the expansion terms are selected.

In order to present the lexical world of a term, several solutions are possible. As a first solution, we characterize the lexical world of a term by a set of “single words" (*also called bag of words*) which are located around the term and present “Noun", “Verb", or “Adjective" parts of speech. As a second solution, the lexical world of a term is presented by co-occurring words like bi-grams of words (i.e., any two adjacent words) or multi-word terms (a few adjacent words, following a pre-defined part-of-speech pattern, which also form a significant term) which are seen around the term. We detail this point in the next section.

Terminological Knowledge

As mentioned above, the lexical worlds can be characterized in different ways: by “single words", “multi-word terms", or “bi-grams of words". According to our experiments, multi-word terms and words are more representative than bi-grams of words. Hence, we create two types of lexical world: (1) consisting of single words and (2) comprising multi-word terms and single words. In order to determine the multi-word terms, we extract the terminology of logs as presented in Chapter 4. This method, adapted to the specific characteristics of logs, extracts multi-word terms according to syntactic patterns in the log files. To choose the *relevant and domain-specific terms*, we also propose a terminology validation and filtering protocol in Chapter 4 . We finally obtain the valid and relevant multi-word terms (e.g., “scan chain", “clock pulse", “design flow") to characterize the lexical worlds.

3.4.1 Query Enrichment by Context Learning

We explain here the first module of our query enrichment approach. For a given question, we initially aim to learn the context of the question and characterize it by its most significant terms. Since these terms represent at best the context of the question, it is expected that passages corresponding to the question share some of these terms regardless of the type of log files.

To identify the context of a question, we lean on the lexical worlds of its keywords. Hence, for every keyword, we look for its lexical world in the training corpus by identifying the small fragment of log files wherein the keyword is located[9]. These fragments are identified based on the recognition of logical units in log files as explained in Chapter 2. Then we characterize the extracted fragments to finally obtain the corresponding lexical worlds of the keyword.

At this stage, we aim at identifying the *most relevant lexical world* among all extracted ones which *presents at best the question context*. Thus, we use Information Retrieval (IR) approach to order the lexical worlds of a question keywords based on their relevance to the question. We use a Vector Space (VS) model. Thus, each document (*here* lexical world) is represented as a vector of values where each value corresponds to a significant term of documents. To calculate these values, also called term indexes, IR systems use an indexing function based on *binary representation*, *Term Frequency* (TF), or *TF-IDF*. Depending on the characteristics of the corpus of logs and our first experiments, we chose the *TF-IDF* to index lexical worlds at this point. *tf-idf* is a statistical weighting function which assesses how important a word is in a document of a corpus [Salton and McGill, 1986]. The values depend on the number of occurrences of words (TF) in a document, and also on the number of documents containing the word (IDF). We detail this measure in Chapter 4, Section 4.3.2.

Once the most correlated lexical worlds are retrieved via IR, a user assesses the relevance of the lexical worlds by choosing the lexical world which presents the question context at best among the retrieved lexical worlds. The chosen lexical world is seen as an explicit relevance feedback. We then identify the *most representative terms* of the chosen lexical world. For this purpose, we use the term indexes (obtained by tf-idf) to measure the importance of them in the context. Finally, we select the *n* terms having the highest tf-idf scores [Salton and Buckley, 1987] as expansion terms.

Since the next phase of query enrichment based on the novel weighting function

9. A given keyword can correspond to few lexical worlds according to its number of occurrences in the corpus.

deserves to be fully developed and studied in detail, we fully devote Section 3.5 in this regard. Therefore, we explain, in the following subsection, how we look for relevant passages once the initial queries are enriched.

3.4.2 Passage Retrieval in Log Files

Here we detail the process of finding relevant passages in the test corpus of log files. First, we segment the logs of the test corpus. Segmentation is performed according to the logical units of log files like data blocks, tables, separating lines, etc. Each segment is seen as a passage of log files potentially containing the answer. The segmentation of log files based on their logical structure is previously developed in Chapter 2.

Second, we enrich the initial query using the relevance feedback in order to make it relevant to all types of log files.

Third, we build an IR system in order to find the relevant passages. For this purpose, we adapted the Lucene search engine[10]. Lucene is a high-performance, full-featured text search engine library written entirely in Java. It offers features like scalable, high-performance indexing, powerful, accurate and Efficient search algorithms, and cross-platform solutions. In addition, [Tellex et al., 2003b] notes that passage retrieval algorithms using Lucene actually achieve a higher performance on average.

Nevertheless, we first need to change the *preprocessing* and *document representation* methods in Lucene to adapt them to the log file characteristics. The special preprocessing of log files is developed in Chapter 4. Once Lucene methods are adapted to the domain characteristics, we use its search engine based on a vectorial model to retrieve the relevant passages.

At this point, we also investigate query creation using initial keywords and expansion terms. Therefore, we need to carefully balance the original query (containing only the initial keywords) and feedback information (expansion terms) because if we over-trust the feedback information, then we may be biased in favour of a particular subset of relevant documents, but under-trusting the feedback information would not take advantage

10. http://lucene.apache.org/

of the feedback. That is why we consider a weight for expanding terms as well as the initial terms. The relevant weights are obtained within some experiments. The details are provided in Section 3.7.

Several Passage Retrieval approaches return a considerable number of candidate passages. Our experiments conducted on real data assert that in more than 80% of cases the relevant passage is located among the three top-ranked passages. A detailed study on Passage Retrieval results and the impact of different parameters like the use of terminological knowledge, number of expansion terms, etc., are provided in Section 3.7.

3.5 How to Find Terms Correlated to Answers

This phase of query expansion is done just before the Passage Retrieval and directly using the *test corpus* (in which we seek relevant passages). At this level, we aim at expanding the query directly using information of the test corpus[11]. Using the test corpus to expand the query makes it impossible to have explicit relevance feedback. Thus, we propose to use pseudo relevance feedback in order to determine the expansion terms at this stage.

In pseudo relevance feedback, the expansion terms are extracted from the top-ranked initially retrieved documents since these documents are supposed to be the most relevant. However, we try to define a new way to determine which terms in a corpus should be selected to expand queries instead of extracting them in top-ranked initially retrieved documents which are blindly considered as relevant.

Therefore, we look for terms which are likely to exist in the relevant passage and are, therefore, related to the query. We thus propose here a *novel and original term weighting function*, called TRQ (Term Relatedness to Query), which gives a score to each term of the test corpus based on its relatedness to the initial query. Therefore, the expansion terms are only selected based on their TRQ scores described in the following paragraphs.

11. In the previous phase, the query is enriched using the information of a training corpus.

3.5.1 Term Relatedness to Query (TRQ) Measure

Firstly, we use the hypothesis presented below to reduce the search spaces, wherein we look for expansion terms in some relevant fragments of the test corpus.

– Hypothesis: The most *correlated terms* to query should exist in a *lexical world* representing the *context* of the question

Based on this hypothesis, for each query keyword, we extract their lexical worlds in the *test corpus*. We note that the system has no information on the logs of the test corpus and relevant passages [12]. We finally obtain a set of lexical worlds, corresponding to the query keywords, which are extracted from the test corpus. These lexical worlds are used as search spaces wherein we seek the terms related to the query.
Our scoring function, TRQ, is then calculated based on the two assumptions:

– The final query must contain *discriminant terms* (i.e., terms which do not exist in several passages) to be efficient and relevant.
– *Most of the relevant query keywords* should be associated with the corresponding *relevant passage* (i.e., the relevant passage contains most of the query keywords)

Firstly, we seek to select discriminative terms. In other words, we look for terms with a very low frequency of occurrence in different lexical worlds. For this, we use the IDF function by considering each lexical world extracted in the previous step as a document and all lexical worlds as a corpus. In this way, we favour terms which exist in one or a very small number of lexical worlds.

Secondly, in order to favour terms which likely exist in the relevant passage, we give another score (*besides idf*) to each term based on the second assumption. For a given term t, this score that we call lwf (Lexical World Frequency) depends on the number of query keywords which are associated with the lexical world wherein the term t exists. This score presents a form of df (Document Frequency) where *a document* corresponds to *all lexical worlds associated* with *a query keyword*. Indeed, by this score, we measure *the importance* of the *lexical world* in which the *given term* is located. This importance is calculated according to the number of query keywords which are associated with the

12. The learning process is performed on logs of the training corpus which are generated by a tool using a vocabulary and structures that differ from the tool generating the logs of the test corpus.

lexical world.

The lwf formula for the term t existing in the lexical world i is calculated as follows:

$$\mathbf{lwf_{ti}} = \frac{1}{\log(\mathbf{K}/\mathbf{K_i})}$$

K is the total number of query keywords and K_i shows the number of query keywords which are associated with the lexical world i. The final score of a term, i.e. TRQ (Term Relatedness to Query), is calculated using the following formula:

$$\mathbf{TRQ(ti)} = \boldsymbol{\alpha} * \mathbf{lwf_{ti}} + (\mathbf{1} - \boldsymbol{\alpha}) * \mathbf{idf_t} \quad \alpha \in [0, 1]$$

According to the experiments, the most relevant value of α is 0.25. This means that we give more weight to IDF and a smaller weight, but which influences the final results to lwf.

Exemple.

We explain, with an example, the process of selection of terms which are likely related to the answer. Supposing Q={W_a, W_b, W_d} is a query enriched by the first module (learning phase) and $\log_b$ is a log file containing seven segments:

$\mathbf{S_1}$={W_a W_k W_m W_b}	$\mathbf{S_4}$={W_a W_c W_e W_q}	$\mathbf{S_7}$={W_b W_c W_k}
$\mathbf{S_2}$={W_d W_k}	$\mathbf{S_5}$={W_b W_e}	
$\mathbf{S_3}$={W_z}	$\mathbf{S_6}$={W_z}	

In this example, we suppose that the border of lexical worlds (*border of the selected fragment of text around a given term*) corresponds to the border of segments (i.e., a lexical world is not bigger than the corresponding segment). Among seven segments, five are associated with terms of Q. Thus, we obtain S_1, S_2, S_4, S_5, S_7 as the set of lexical worlds of question keywords. The following lists show the lexical worlds associated with each keyword of the question[13].

W_a:{S_1, S_4} W_b:{S_1, S_5, S_7} W_d:{S_2}

13. Since a word can be used in different contexts (i.e., different fragments of document), it can be associated with several lexical worlds.

Here, for instance, the word W_a in the query Q is associated with two lexical worlds (S_1 and S_4). The idf score of the word W_k, for example, is equal to $log(\frac{5}{3}) = 0.22$ because the word W_k exists in three lexical worlds (S_1, S_2 and S_7) among five. The value of lwf for the word W_k in the segment S_1 is calculated as follows: $lwf_{k,1} = \frac{1}{\log(\frac{3}{2})} = 5.8$. Indeed, two words in the query Q (i.e. W_a and W_b) are associated with the lexical world S_1 (the lexical world in which the word W_k is located). We note that, for a given term, the value of lwf depends on the lexical world in which the given term is located. For example, the value of lwf for the word W_k located in segment S_2 is equal to $lwf_{2,2} = \frac{1}{\log(\frac{3}{1})} = 2.1$ as there is just one keyword of the query Q associated with S_2. This means that W_k located in segment S_2 is less significant (less related to the query) than when it is located in segment S_1.

3.5.2 Considering Terms Dependency in the TRQ measure

Another factor to consider in the selection of terms related to a query, is how they are correlated to the query keywords. For this purpose, we assess the *tendency* of terms to appear *close* to keywords of the initial query. In other words, we seek to calculate the *dependence* of *terms* to the *keywords* of the question in the studied context. To calculate the dependence of terms, several measures can be applied: Mutual Information [Guiasu, 1977], Cube Mutual Information [Daille, 1996b], and Dice coefficient [van Rijsbergen, 1979]. We choose the "Dice" coefficient for this purpose as this statistical measure has good performance for text mining tasks [Roche and Kodratoff, 2009]. Also, as noted in [Tellex et al., 2003b], in general, a scoring function based on how close keywords appear to each other is common among passage retrieval algorithms. Dice value makes it enable to calculate how two terms are dependent based on their occurrence distance in a corpus.

For two terms X and Y, the Dice coefficient is defined as twice the number of times X and Y appear together over the sum of the total number of times that each one appears in the corpus.

$$Dice(X,Y) = \frac{2 * |(X,Y)|}{|X| + |Y|}$$

Since we would like to measure the *dependence of a given term to a query rather than to a keyword*, we calculate the *sum of Dice values for each pair of the term and a keyword of* *the query*. Thus, we obtain an extended version of the Dice coefficient which measures the dependence of a term T to a query Q.

$$Dice_{ext}(T,Q) = \sum_{i=1}^{|K_Q|} \frac{2 * |(T, k_i)|}{|T| + |k_i|}$$

The $|K_Q|$ presents the number of keywords in the query Q and k_i is a keyword of Q.

We discuss here the notion of appearing together in the Dice coefficient. For us, $| \, (T, k_i) \, |$, i.e., number of times where T and k_i occur together, corresponds to the number of times where T and k_i are located in the same line in the corpus of logs. Although we agree that the distance between T and k_i can be taken into account if necessary, but in log files we look only for the close occurrence of the two terms without limiting it to a predefined distance in the same line, which does not seem relevant. Moreover, we note that we could extend the "same line" constraint to a window of lines. This means that T and k_i occur together if there are just a predefined number of lines between them.

We finally extend the previously presented TRQ measure by integrating the extended Dice value ($Dice_{ext}$) into it. Thereby, we obtain the final extended TRQ measure which is formulated as following :

$$\mathbf{TRQ_{ext}(ti, Q)} = \boldsymbol{\alpha} * (\mathbf{lwf_{ti}}) + (\mathbf{1} - \boldsymbol{\alpha}) * \mathbf{idf_t} + Dice_{ext}(t, Q)$$

Thus, the $\mathbf{TRQ_{ext}}$ measure also takes into account the dependency of terms on a given query in the studied context. Note that, by calculating only the simple version of TRQ, we have many terms with equal TRQ values. This situation can happen for terms situated in passages wherein there is the same number of keywords. In such situation, taking the extended Dice value $Dice_{ext}$ makes it enable to distinguish terms who are more likely to exist in answers. We note that we could not use only Dice measure since there are terms dependent on queries, which do not exist in relevant passages. In order to best identify the expanding terms, which are likely existing in relevant passages, we have to consider all parameters of TRQ_{ext}. The assumptions based on which we have defined TRQ measure ensure that expansion terms are selected in passages which are likely the relevant ones and the terms likely exist in answers.

3.5.3 Term Selection using the TRQ measure

Once the TRQ_{ext} score of all terms of lexical worlds is calculated, we identify the *n highest scores* and select the terms having these scores. Note that, for improving the *performance* of our query expansion approach, we first calculate the simple TRQ of terms (i.e., without calculating extended Dice values). Then, we process to calculate the extended Dice values only for n terms having the highest simple TRQ score.

Once we ranked the terms based on their TRQ_{ext} scores, we have to decide about how to choose the expansion terms. According to our experiments, the selection of terms depends on the occurrence frequency of the answer in the corpus. In other words, it depends on if there are more than one relevant passage in the corpus for a given question. In the context of log files, as there are rarely more than one relevant passage for a question, we first give a score to each passage based on the TRQ_{ext} score of terms that it contains. This score is calculated as the sum of TRQ_{ext} values of the passage terms. Then, although there are only one relevant passage, we select the m top-ranked passages to make the system tolerable to the potential unexpectedness. The expansion terms are finally selected in the m top-ranked passages based on the highest TRQ_{ext} values. The system automatically integrates the m top-ranked terms into the query in an autonomous mode. The enriched query will be used in passage retrieval.

In a context where there are more than one relevant passage, we need to adapt some part of TRQ score (i.e., calculation of idf) as well as how we select the terms. This issue is developed in detail in Section 3.6 where we discuss how to adapt our approach to the context of open domains.

In the selection of expansion terms, there are some parameters which can impact the relevance of selected terms. The value of these parameters like the use of terminological knowledge, the number of expansion terms, or the weight of expansion terms in the final queries are determined within an exhaustive experimental protocol whose results are presented in Section 3.7.

3.6 Application of the Query Expansion Approach in Open Domains

Despite the fact that our approach is designed by default for the restricted domain presented before (i.e., CAD and EAD log files), here we study its application to open domains. That reveals how we can adapt our query expansion methods in order to also work well on general fields.

We present here a brief introduction of the general documents used in this study. The test results are presented and discussed in section 3.7. We use a corpus of general documents which are used in TREC'04 Novelty Track. This data is designed to investigate the system abilities to locate relevant information within a set of documents concerning a TREC topic. Given a TREC topic and an ordered list of documents, systems must find relevant and novel sentences that should be returned to the user from this set. This task integrates aspects of passage retrieval and information filtering [National and Soboroff, 2004]. Some topics are about air flight disasters, ship cloning, political opinions, sport events, etc.

This corpus differs in some principal aspects from the corpus of log files, which requires some adaptation of the query expansion approach. Firstly, the general documents are written in a natural language and they comply with its grammars. Thus, there are more relevant semantic and syntactic relations between the terms of the corpus. Moreover, there are no meaningful special characters or terms containing such characters. Secondly, the document structures and vocabulary are not different in the open domain corpus. This means that it is less challenging to reformulate the relevant expanded queries. In open domains, contrary to the restricted domains, it is not expected that there will be any new documents which would likely have the structures and vocabulary that differ significantly from those of training corpus. Thirdly, in the open domain corpus, we have more than one relevant passage for a given query. Thus, in the explicit relevance feedback phase, we can have more than one relevant passage wherein we have to look for the expansion terms. That means that we need to change how we select the expansion terms in the training phase.

According to these differences, we adapt some parts of our approach. To consider the semantic and syntactic relations in the general documents, we study the impact of using terminological knowledge in query expansion in open domains. For this purpose, we compare the test results obtained by using terminological knowledge in the restricted domain with results obtained in the open domain application.

Based on the second difference, the impact of the second query expansion phase is evaluated in the experiments. Since vocabulary changes in the open domain test corpus are not expected to be significant, the impact of the second query expansion phase should not be as same as its impact in the restricted domain. This issue is also studied in the experiments in Section 3.7.

Finally, the last difference requires some changes in expansion term selection in the *first phase of enrichment* (i.e., context learning). As we can have more than one relevant passage in the training phase, we change the term selection method at this stage as follows. By default, the expansion terms are extracted in one relevant passage (identified within the learning process) as in the corpus of log files there is only one relevant passage for a given question. In the case of open domains, the user can choice more than one passage as the relevant passages in the phase of training. Thus, we first select the most representative terms of each selected relevant passage. Then, to select the most relevant expansion terms, we seek terms which are seen in most of the relevant passages. Thus, terms having a higher frequency of occurrence in the relevant passages are selected for expansion.

Calculation of the IDF score in the TRQ measure is also slightly changed to take into account the fact that there is surely more than one relevant passage in the open domain corpus. In fact, in the case of log files, as there is only one relevant document, we calculate the IDF score by considering each extracted lexical world as a document and their collection as the corpus. However, in the case of open domains, there are usually few lexical worlds which are likely all relevant. As by the IDF score we favour terms having less occurrence in different documents, we can eliminate representative terms of *relevant* lexical worlds because they likely occur in *all relevant* lexical worlds. That is why we merge all extracted lexical worlds to create a single document. We calculate the IDF, while this created document as well as other segments of the test corpus constitute a new corpus.

As by merging the lexical worlds into a single document, we increase the occurrence frequency of representative terms in the document, we highlight the IDF score by the TF score of terms in the new document built by merging the lexical worlds. The impact of these changes and the required configurations in the open domain case are evaluated in the experiments presented in the next section.

3.7 Experiments

Here we present the results of tests performed to evaluate the performance of our approach. We also determine the best values for every parameter. The tests are performed in two principal categories. Firstly, we present our experiments in the restricted domain of log files. Then the same tests are carried out to evaluate application of the approach in an open domain (TREC data). As explained in Section 3.6, the query expansion approach is adapted to some characteristics of open domain texts which do not exist in the restricted domain of log files of EDA.

We study different values for only three parameters of our approach: (1) *Usage of terminological knowledge*, (2) *Weight of expansion terms*, and (3) *Number of expansion terms*. Our test protocol allows us to determine the best values for every parameter in each category. Moreover, we compare the impact of each parameter in a category to its impact in the other one.

3.7.1 Experiments on Log Files

We test the performance of our approach on a corpus of log files from the real industrial world (*data from Satin Technologies*). The questions, expressed in natural language, are all extracted from a *standard check-list* designed by societies like IEEE. The choice of questions rises some difficulties as we require them to be extracted from real industrial quality verification check-list. In the same time, we have to have access to real log files which contain the answers of selected questions. We finally obtained 36 questions which are used in real industrial applications.

Log files are segmented according to their structures (like blank lines, tables, data blocks) by using the approach proposed in Chapter 2. Each segment is potentially a relevant passage. Note that for a given question, there is only one relevant passage (*segment*) in the corpus. The relevance of passages is evaluated according to whether the final answer is located in the passage.

The test corpus that we use for the experiments contains 625 segments and is about 950 KB. Each segment consists of approximately 150 words. The training corpus used in the context learning phase consists of logs reporting the same information as logs of the test corpus, but generated by a totally different tool – *thus different vocabulary and segmentation*. For a given question, the initial query is obtained on the basis of the question keywords.

In order to evaluate the performance of our approach in different conditions, we use the *Mean Reciprocal answer Rank* (MRR) used in TREC as a performance evaluation measure [Voorhees, 1999]. This measure takes into account the rank of the correct answer among the ranked list of candidate answers.

$$MRR = \frac{1}{nb(Question)} \sum_{i=1}^{nb(Question)} \frac{1}{rank(answer)}$$

We also calculate the *Recall* as the *number of questions* for which the *relevant passage* is found among the *top five retrieved passages*.

Moreover, we demonstrate by $P(n)$ the *percentage of questions* for which the *relevant passage* is *ranked as n* among the retrieved candidate passages as possibilities. In the following sections, we show the results for the first three ranks.

Firstly, we present the results of different tests using the different configurations. Once we obtain the best configuration, we compare the Passage Retrieval performance using enriched queries with the performance of passage retrieval using the initial queries (not enriched). That shows how our query enrichment approach improves the relevancy of the initial queries.

Usage of Terminological Knowledge

In this test, we use the training and test corpus described previously. First, we do not use the domain terminological knowledge in the query expansion phases. This means that we do not extract the domain terms and thus there are not any multi-word terms among the expansion candidate terms. Second, we conduct the same test, but by using the terminological knowledge of the domain. In both tests, all other parameters remain unchanged. As the results show (see Table 3.2 [b]), we have a performance gain by

[a]

	Explicit FB	Pseudo FB
P(1)	29	24
P(2)	0	7
P(3)	3	3
MRR	**0.84**	**0.80**
Recall	**0.88**	**1**

[b]

	Explicit FB	Pseudo FB
P(1)	29	26
P(2)	1	7
P(3)	3	3
MRR	**0.84**	**0.85**
Recall	**0.91**	**1**

Table 3.2: Performance of passage retrieval in log files while **(a)** the domain terminological knowledge is not used, **(b)** the terminological knowledge is used.

considering the terminological knowledge of the domain. Multi-word terms are more significant and discriminative than simple terms. Hence extracting multi-word terms to characterize the context of questions (lexical worlds) helps find more significant expansion terms. Although the MMR value obtained by the queries expanded in the first phase is close to the MMR value obtained using the queries also expanded in the second phase, we observe that the second query expansion phase using the TRQ measure helps improve the Recall of Passage Retrieval. Hence, by using the second query enrichment phase, all relevant passages are situated among the five top-ranked retrieved passages. According to this test, we use the terminological knowledge in all following tests.

Weight of Expansion Terms

Here we present some tests to evaluate the best value for the weight of expansion terms in the enriched queries. In fact, we give the value 1 as weight to the initial keywords of queries and a fraction of one to expansion terms. We choose three values: 0.2, 0.5, 0.7. Each test is performed using one of the values as the weight of expansion terms while the other parameters remain the same. Table 3.3 presents the test results regarding the different expansion term weight values.

[a]

	Explicit FB	Pseudo FB
P(1)	29	27
P(2)	1	7
P(3)	3	2
MRR	**0.85**	**0.87**
Recall	**0.94**	**1**

[b]

	Explicit FB	Pseudo FB
P(1)	29	26
P(2)	1	7
P(3)	3	3
MRR	**0.84**	**0.85**
Recall	**0.91**	**1**

[c]

	Explicit FB	Pseudo FB
P(1)	27	24
P(2)	3	6
P(3)	3	6
MRR	**0.82**	**0.81**
Recall	**0.94**	**1**

Table 3.3: **(a)** weight of expansion terms = 0.2, **(b)** weight of expansion terms = 0.5, **(c)** weight of expansion terms = 0.7. Tests performed on log files.

According to the results, we obtain the best performance by using 0.2 as the weight of expansion terms in the enriched queries. By increasing the weight of expansion terms, the MRR value decreases slightly. Therefore we use 0.2 in all of the following terms as the weight of expansion terms.

Number of Expansion Terms

We try to find the best number of expansion terms to include in the initial queries. An irrelevant number of terms can bias the performance. Actually, including all expansion candidate terms is not relevant because this can bias the relevance of queries and significantly decrease the importance of the initial keywords. A few numbers of terms can be irrelevant because we may add insufficient amount of information to the initial queries. Thus, we select three values (3,5,7) as the number of expansion terms to integrate into initial queries. We present the results in Table 3.4.

As shown in this table, the best result is obtained while we select only the *three* most relevant expansion terms. As we observed in the case of the weight of expansion terms,

[a]

	Explicit FB	Pseudo FB
P(1)	29	27
P(2)	1	7
P(3)	3	2
MRR	**0.85**	**0.87**
Recall	**0.94**	**1**

[b]

	Explicit FB	Pseudo FB
P(1)	28	25
P(2)	2	7
P(3)	3	2
MRR	**0.83**	**0.82**
Recall	**0.94**	**1**

[c]

	Explicit FB	Pseudo FB
P(1)	28	25
P(2)	2	5
P(3)	1	2
MRR	**0.82**	**0.81**
Recall	**0.91**	**1**

Table 3.4: **(a)** number of expansion terms = $\mathbf{3}$, **(b)** number of expansion terms = $\mathbf{5}$, **(c)** number of expansion terms = $\mathbf{7}$. Tests performed on log files.

by increasing the number of expansion terms, the relevance of the expanded queries decreases.

Initial Queries Vs. Expanded Queries

We aim at evaluating the performance gain while we perform Passage Retrieval using the expanded queries. We thus perform the Passage Retrieval in log files once using the initial queries (not enriched) and once using the enriched ones. We use the best values obtained in the previous tests for the corresponding parameters of our query expansion approach. Table 3.5 presents the results for both tests. As shown in this table, by using

	Initial Queries	Expanded Queries
P(1)	24	27
P(2)	2	7
P(3)	2	2
MRR	**0.71**	**0.87**
Recall	**0.80**	**1**

Table 3.5: Performance of Passage Retrieval in log files obtained by using the **non-enriched queries** (initial queries) vs. performance obtained by using **enriched queries**

the non-enriched queries, we obtain an MRR value equal to 0.71 in the best conditions.

While by enriching the initial queries using our approach, the MRR significantly improves and reaches 0.87 in the best conditions.
According to the results, in the best configuration and by *using our query enrichment approach*, the relevant passage is ranked in 75% of cases as the first passage among the candidate passages returned by the system. Moreover, in 100% of cases, the relevant passage is located (ranked) among the five top-ranked passages returned by the system when there are about 650 passages in the corpus.
In the next section, we evaluate our approach on documents from general domains.

3.7.2 Experiments on TREC Data

In this evaluation, we aim at studying the performance of our query expansion approach in general domains. For this purpose, we use the documents used in the TREC'04 Novelty Track[14]. In this corpus, there are several files each one containing several small documents. For a given topic, there are some corresponding documents in every file. That is why we consider the corpus as a collection of documents regardless of their physical location in several files. Since every document contains between 10 and 20 lines, we consider them as passages. In the corpus, we have 1420 documents (passages) and there are 50 search topics. Each topic, like "Find opinions about the partial birth abortion ban", presents a query. For a given topic, there are on average 10 relevant documents (passages).

The test performances are calculated using the 3-fold cross validation. We follow the same test procedure that we used in the case of log files.

Usage of Terminological Knowledge

We focus on studying the impact of using terminological knowledge within our query expansion approach in the open domains. Table 3.6 presents the Passage Retrieval results based on the usage and non usage of terminological knowledge. The MMR value is equal to 0.91 in the case of non usage of terminological knowledge, as compared to 0.99 in the case of using this knowledge. As the results show, we have a considerable performance gain when we use the terminological knowledge in our query expansion

14. http://trec.nist.gov/data/t13_novelty.html

[a]

	Explicit FB	Pseudo FB
P(1)	45	45
P(2)	1	2
P(3)	0	1
MRR	**0.91**	**0.92**
Recall	**0.96**	**1**

[b]

	Explicit FB	Pseudo FB
P(1)	49	49
P(2)	1	1
P(3)	0	0
MRR	**0.99**	**0.99**
Recall	**1**	**1**

Table 3.6: Performance of Passage Retrieval in TREC data (open domain) while **(a)** the domain terminological knowledge is not used, **(b)** the terminological knowledge is used.

approach. By comparing these results to those obtained on log files (cf. Table 3.2), we observe that the usage of terminological knowledge is more significant on open domains. It is due to the fact that there are sufficient significant multi-word terms in open domain texts. However, in log files, multi-word terms are not numerous, and their extraction and validation are more challenging than their extraction in open domains. We note at the same time that the usage of terminological knowledge in the case of log files is nevertheless relevant and helps to improve the query expansion performance.

Weight of Expansion Terms

We also perform some tests to evaluate the best values for the weight of expansion terms in the context of open domains. As in the case of log files, we chose three values: 0.2, 0.5, 0.7. Each test is performed using one of the values as the weight of expansion terms while the other parameters remain the same. We show the results obtained in Table 3.7. According to the results, the weight of expansion terms does not considerably influence the relevance of the expanded queries in the TREC data context. We obtain the best results while the weight value is set at 0.2.

Number of Expansion Terms

We now aim to determine the best number of expansion terms to include in the initial queries in the context of open domains. We again select three values (3,5,7) as the number of expansion terms. Table 3.8 presents the obtained results based on the number of expansion terms. As shown in Table 3.8, the performance change based on the different number of expansion terms is not considerable. However, the best result is

[a]

	Explicit FB	Pseudo FB
P(1)	49	49
P(2)	0	0
P(3)	0	1
MRR	**0.98**	**0.98**
Recall	**0.98**	**1**

[b]

	Explicit FB	Pseudo FB
P(1)	49	49
P(2)	1	1
P(3)	0	0
MRR	**0.99**	**0.99**
Recall	**1**	**1**

[c]

	Explicit FB	Pseudo FB
P(1)	49	49
P(2)	0	0
P(3)	0	1
MRR	**0.98**	**0.98**
Recall	**0.98**	**1**

Table 3.7: **(a)** weight of expansion terms = 0.2, **(b)** weight of expansion terms = 0.5, **(c)** weight of expansion terms = 0.7. Tests performed on TREC data.

[a]

	Explicit FB	Pseudo FB
P(1)	49	49
P(2)	1	1
P(3)	0	0
MRR	**0.99**	**0.99**
Recall	**1**	**1**

[b]

	Explicit FB	Pseudo FB
P(1)	48	48
P(2)	0	1
P(3)	0	0
MRR	**0.96**	**0.97**
Recall	**0.96**	**0.98**

[c]

	Explicit FB	Pseudo FB
P(1)	47	48
P(2)	0	0
P(3)	1	0
MRR	**0.94**	**0.96**
Recall	**0.96**	**0.96**

Table 3.8: **(a)** number of expansion terms = 3, **(b)** number of expansion terms = 5, **(c)** number of expansion terms = 7. Tests performed on TREC Data.

obtained when we select only the *three* most relevant expansion terms. By increasing the number of expansion terms, the relevance of the expanded queries slightly decreases.

Initial Queries Vs. Expanded Queries

Here we aim to evaluate the performance gain while we perform Passage Retrieval using expanded queries in the open domain context. As we did in the case of log files, we perform Passage Retrieval once using the expanded queries and once using the initial ones. Table 3.9 presents the results of both tests. According to the results, we

	Initial Queries	Expanded Queries
P(1)	37	49
P(2)	2	1
P(3)	1	0
MRR	**0.75**	**0.99**
Recall	**0.80**	**1**

Table 3.9: Performance of passage retrieval in TREC data obtained by using the **not enriched queries** (initial queries) vs. performance obtained by using the **enriched queries.**

obtain an MRR value equal to 0.75 while we use the initial queries. When expanding the initial queries using our approach, we obtain a significant improvement in terms of MRR value (equal to 0.99). Moreover, in such configurations, the Recall of system is equal to 1, which means that all relevant passages are ranked among the first five.

By *using our query expansion approach*, the relevant passage is ranked in 98% of cases as the first passage among the candidate passages returned by the system. Moreover, in 100% of cases, the relevant passage is located (ranked) among the five top-ranked passages returned by the system when there are about 490 passages in the test corpus.

3.8 Discussions

In this chapter, we investigated the problem of locating information in all types of log files. For this purpose, we proposed an approach to retrieve relevant passages of log files which contain the requested information. A passage is actually a segment of log files, whereas segments are obtained after splitting log files by our segmentation approach presented in Chapter 2. The main problem in passage retrieval in log files was due to the fact that log files are multi-source and multi-vocabulary data. This results in mismatch vocabularies issue, i.e., the gap between vocabulary of queries and those of log files. That is why we proposed an approach of query expansion based on two phases of relevance feedback.

In Section 4.2, we studied the existing work in the domains of passage retrieval and query expansion. We presented three main directions: NLP & statistic methods, global query expansion based methods, and methods based on the local query expansion. We first argued that the NLP based methods are irrelevant in the context of log files because of some points notably the lack of information redundancy, existing of several technical and alphanumeric terms for which the use of syntactic or semantic variants is meaningless, and lack of external knowledge base like ontologies or dictionaries. The global query expansion methods also suffer from some difficulties like the semantic ambiguity and lack of semantic resources to determine semantic relations. Local query expansion consists in using relevance feedback. In the case of explicit relevance feedback, we first noticed that indexing the whole training corpus for the initial retrieval can be time-consuming for industrial usage. Second, the initially retrieved passages are long enough to contain terms that are not really correlated to the query terms. There are also some difficulties in using classic methods of blind relevance feedback. Indeed, in log files, the top-ranked initially retrieved documents are not always relevant. Thus, considering the top-ranked initially retrieved documents as relevant, which is the principal of blind relevance feedback, reveals to be questionable.

In Section 3.4, we proposed our query expansion approach based on two new methods of relevance feedback. Our query expansion approach is based on a *context learning* process and is associated with an *original term weighting function*. Our protocol of context learning is designed to determine the context of a given question, which results in reducing the search space during the initial retrieval. We also try to define a new way to determine the terms that should be selected to expand queries instead of extracting them in top-ranked initially retrieved documents which are by default considered as relevant in the case of blind relevance feedback. Thus, we proposed a novel and original term weighting function, called TRQ (Term Relatedness to Query), which gives a score to each term of the test corpus based on its relatedness to the initial query. Therefore, the expansion terms were only selected based on their TRQ scores. The two processes of the query expansion based on the context learning and the TRQ measure are respectively developed in Section 3.4.1 and 3.5. We also explained the process of passage retrieval using the expanded queries in Section 3.4.2.

In Section 3.6, we studied the application of our query expansion approach to open domains, however it was designed by default for the restricted domain like EAD log files. We developed, in this section, the different points that should be adapted. The main issue to consider was the redundancy of answer in open domains.

We evaluated our approach in Section 3.7 by using the real industrial log files. The findings showed that by using our query expansion approach in the passage retrieval process in log files we improved the performance by 22% in terms of MRR value and 25% in terms of Recall. These results prove the efficiency of our query expansion approach based on a context learning phase and our original TRQ measure.

However, our approach is not designed by default to work in open domains but after the adaptations mentioned in Section 3.6, we highlight that our query expansion helps to also obtain more relevant queries in the open domain. According to the results, we obtain 46% performance gain in terms of MRR value and 25% performance gain in terms of Recall.

During the passage retrieval and query expansion, the use of terminological knowledge revealed to be relevant. Characterizing the context of questions using multi-word terms was more efficient than only using the simple terms. Using multi-world terms among the expansion terms helped to improve further the relevance of obtained queries. The use of terminological knowledge in this domain proves to be highly relevant and can significantly improve the results. This point motivated us to investigate the extraction of terminological knowledge in the log files. According to the specificities of log files, it was revealed no trivial task. In the next chapter, we present our work in the domain of terminological knowledge extraction in log files.

Chapter

4

Adding Terminological Knowledge into the IE System

Doubt is the key to knowledge.

Persian Proverb

Preamble

In this chapter we first discuss the use of terminological knowledge in our passage retrieval and query expansion approaches. Then, we study how to extract the EDA domain-specific terminology in log files. For this purpose, we introduce our approach Exterlog *to extract terminology from log files. We detail how it deals with the specific features of such textual data. The performance is emphasized by favoring the most relevant terms of the domain based on a scoring function which uses a* Web *and* context *based measure. The experiments show that* Exterlog *is a well-adapted approach for terminology extraction from log files. We have published* Exterlog *and our term validation protocol respectively in [*Saneifar et al.*, 2009] and [*Saneifar et al.*, 2011a].*

Contents

4.1 Introduction

Adding linguistic knowledge to Information Retrieval process can improve the retrieval performance. Methods for integrating linguistic content within information retrieval activities are receiving a growing attention [Moldovan and Mihalcea, 2000]. In our context, we have observed during experiments a significant improvement in performance of passage retrieval as well as query expansion by using the domain terminological knowledge. Using the EDA domain-specific terms as textual features to characterize documents (e.g., passages, lexical words) provides a more relevant presentation of the documents. Thus, we focus in this chapter on the issue of how to acquire the terminological knowledge specific to the EDA domain.

Terminology is the sum of the terms which identify a specific topic. According to [Witschel, 2005], the notion of terminology is defined in the ISO 1087 as "Set of terms representing the system of concepts of a particular subject field". In this way, terminology extraction, also called terminology mining, term extraction, or term recognition is a subtask of information extraction and natural language processing which aims at extracting terminology from a text.

When working on specialized languages and specific domains, terminology plays a crucial role as it aims at describing and organizing the knowledge of the domain through the concepts, and their lexical realizations, that are used [Déjean et al., 2005]. Many terminology engineering processes involve the task of automatic terminology extraction: Before the terminology of a given domain can be modelled, organised or standardised, important concepts (or terms) of this domain have to be identified and fed into terminological databases [Witschel, 2005]. Furthermore, terminology extraction is a very useful starting point for semantic similarity or knowledge management.

Application of domain-specific terminology extraction is studied in several other fields notably domain ontology construction [Kietz et al., 2000], information retrieval and information extraction [Yangarber et al., 2000]. Thus, developing an automatic domain terminology construction method has been arising many interests in different research domains.

Regarding our work, in the phase of passage retrieval, we have introduced the notion of lexical world (see Section 3.4) which is used within our query expansion approach. The lexical worlds are used to determine the context of questions. Each lexical world, which is a text chunk, is represented in our approach with a feature vector [Salton and Buckley, 1987]. Features are linguistic units like words which are sought in the lexical worlds that describe the contents of a given text briefly but meaningfully. In other words, we characterize lexical worlds by means of their linguistic features. Although we can simply use only single words sought in a lexical world as its features, it is more relevant to define more specific features like multi-word terms. As mentioned previously, we obtain the better results while the lexical worlds are characterized by multi-word terms besides the single words. This issue highly motivates us to construct this domain ontology in order to better determine the relevant multi-word terms.

Moreover, using the domain-specific terms as index term in Information Retrieval systems is revealed to improve the retrieval performance. An "index term" is defined as a "word which describes the contents of a document" [Knorz, 1991]. This indicates that index terms are used for representing contents of specific documents. An index term should also help to distinguish a document from others [Witschel, 2005].

We use the domain-specific terminological knowledge to better determine the features of the log files to be used as index terms. The use of multi-word terms, sought in log files, during indexation (for passage retrieval) helps to distinguish documents more accurately. The relevant choice of corpus features has a marked impact on the passage retrieval results.

We note that the obtained terminological knowledge will also serve as a starting point to compiling dictionaries or even to create the EDA domain ontology in our future work. In fact, in order to build such an ontology, we first have to identify the domain terms which will be considered as instances of the ontology.

In this chapter, we hence focus particularly at exploring the lexical structure of log files in order to extract the EDA domain-specific terms. Here, we introduce our approach, named Exterlog (EXtraction of TERminology from LOGs), to extract the terminology of log files. Our approach consists in using the syntactic methods as well as statistic ones. We study within our approach the relevance of two main methods of terminology extraction. These methods are based on the extraction of co-occurrences with and without the use of syntactic patterns. Moreover, in order to automatically validate the relevant candidate terms, we present a method to filter the extracted terms based on a ranking function.

This chapter is organised as follows. We discuss the related work in the domain of terminology extraction in Section 4.2 . Our Exterlog approach is developed in Section 4.3 along with our term filtering method. Section 4.4 describes and compares the various experiments that we performed to extract terms from the log files and to evaluate the performance of Exterlog.

4.2 Related Work

The extraction of domain terminology from textual data is an essential task to establish specialized dictionaries of specific domains [Roche et al., 2004]. The extraction of co-occurring words is an important step in identifying terms. To identify co-occurrences, some approaches like [Penas et al., 2001] are based on syntactic techniques which initially rely on part-of-speech tagging. Candidate terms are then extracted using syntactic patterns (*e.g.* adjective-noun, noun-noun). Part-of-speech (POS) tagging (also called grammatical tagging) is a NLP method used to analyse text files and annotate words based on their grammatical roles. In the same category, we have also Syntex, proposed by [Bourigault and Fabre, 2000], which performs syntactic analysis of texts to identify nouns, verbs, adjectives, adverbs, the noun phrases, and verbal phrases. It analyses the text by applying syntactic rules to extract terms. Defined rules and grammar are also used by [David and Plante, 1990] to extract nominal terms as well as to evaluate them. Exit, introduced by [Roche et al., 2004], is an iterative approach that finds nominal and verbal terms in an incremental way. A term found in an iteration is used in the next one to find more complex terms.

In [Déjean et al., 2005], authors present their work on extraction of bilingual lexicon (English and German) from parallel corpora in the medical domain. The extracted lexicons are semi-automatically used to enrich mono- or bilingual thesauri. In [Déjean et al., 2005], the main focus is on the extraction of lexicon from comparable corpora. The authors argue that their approach is relevant to the medical domain as there are bilingual thesauri in this domain. In order to evaluate the extracted lexicons, they manually extracted a reference lexicon comprising 1,800 translation pairs from the studied corpus. About 1,200 pairs are then reserved for estimating the mixture weights, and 600 pairs for the evaluation. The results are averaged over 10 different such splits.

[Dorji et al., 2011] present a methodology that uses both statistical and linguistic methods to extract and select relevant compound as well as single Field Associated (FA) Terms from domain-specific corpora. An FA Term is defined as the minimum word or phrase that serves to identify a particular field. They use specially developed POS patterns to extract FA Term candidates from domain-specific corpora using a sliding window of ten words. Relevant FA Terms are then selected by corpora comparison and using a unique series of statistical formulae based on tf-idf.

Machine learning methods based on Hidden Markov Models (HMMs) are used in [Collier et al., 2002] to extract terminology in the field of molecular biology. The extraction of terms are performed according to examples that have been marked up by a domain expert in a corpus of abstracts taken from a controlled search of the Medline database.

Some approaches try to extract the collocations in a fixed size window (*e.g. five words*) based on lexical dependency of words. Collocations are linguistic phenomena that occur when two or more words appear together more often than by chance and whose meaning often cannot be inferred from the meanings of its parts [Petrović et al., 2010]. Dekang Lin notes in [Lin, 1998] that extracted words as collocation in a fixed size window may not be directly correlated. Xtract, a terminology extraction system which identifies lexical relations in the large corpus of English texts, avoids this problem by considering the relative positions of co-occurrences [Smadja, 1993]. in Xtract, pairwise lexical relations are first retrieved using only statistical information. After identification of multiple-word combinations and complex expression, by using the parsing and statistic technique, the found collocations are filtered.

More general than a collocation is the term word n-gram which denotes any sequence of n words. Extracting collocations usually proceeds by assigning each candidate n-gram a numeric value indicating how strongly the words within the n-gram are associated with each other [Petrović et al., 2010]. The higher this value, the more likely that the n-gram is a collocation. The functions used to assign these values are called lexical association measures. The most known measures are those used in Information theory like Information Mutual and Dice value [Pecina and Schlesinger, 2006]. [Petrović et al., 2010] focus on extending these measures to make them suitable for extracting longer collocations than bi-grams. Bi-grams are also used in [meng Tan et al., 2002] as index-term to improve the performance of the text classification.

Finally, in order to evaluate the adequacy of candidate terms, statistical methods are generally associated with syntactic approaches [Daille, 2003]. These methods are based on statistical measures such as information gain to validate an extracted candidate as a term. Among these measures, the occurrence frequency of candidates is a basic notion.

Discussing the background methods

In the domain of terminology extraction, most of approaches are based on a combination of some main methods like use of syntactic pattern or statistic measures. Table 4.1 presents these main methods as well as approaches (previously described) which use a combination of them to extract terms.

Many studies compare different techniques of terminology extraction and their performances. But most of these studies are tested on classical texts written in a natural

	Syntactic	Statistic	Lexical dependency	Machine learning
[Bourigault and Fabre, 2000]	√			
[Penas et al., 2001]	√			
[Smadja, 1993]	√	√	√	
[Lin, 1998]	√		√	
[Collier et al., 2002]				√
[Roche et al., 2004]	√	√		
[Petrović et al., 2010]		√		
[Dorji et al., 2011]	√	√		

Table 4.1: Main methods and approaches in Terminology Extraction

language. Most of the corpus used in the experiments of these approaches are consistently structured. Moreover, this textual data complies with NL grammar. However, in our context, due to the characteristics of logs, these methods have to be adapted to ensure that they are relevant for log files.

For instance, as we have previously seen, in the context of log files, there are some difficulties and limitations for applying grammatical tagging and hence using the syntactic pattern on such textual data. Indeed, the classic techniques of POS tagging are normally developed and trained using texts written in a standard natural language, such as journals. They are hence based on standard grammar of natural language in order to determine the grammatical role of words. For instance, they consider that a sentence ends with a full-stop while this is not the case in the log files that we handle. More specifically, in these log files, sentences and paragraphs are not always well structured.

Moreover, there are also some difficulties in using the statistic methods to evaluate and validate the candidate terms. The statistical methods used in classical term extraction methods cannot be applied to log files as they are. Indeed, statistical approaches can cope with high frequency terms, but tend to miss low frequency ones [Evans and Zhai, 1996]. Information is seldom redundant according to the characteristics of log files. Therefore, the domain terms often have very low occurrence frequency. Thus, in our context, we cannot use classic statistical measures which are often relevant to validate the frequent terms.

In the next section, we develop our approach of terminology extraction from log files. Within our approach, we explain how to pre-process the log files in order to prepare them to apply NLP methods. We describe how to adapt a POS tagger to the characteristics of log files. Then, we use a syntactic-based method to extract candidate terms. We finally propose an extended statistic measure and a term evaluation protocol by considering the specificities of log files. Using these adapted methods and the proposed evaluation protocol we overcome the difficulties seen in the extraction of terms in log file corpus.

4.3 Exterlog: EXtraction of TERminology from LOGs

Our approach, Exterlog, consists of two main phases:

– Extraction of terms

– Filtering relevant terms

Figure 4.1 shows the main architecture of our approach. In the first phase, i.e., Extraction of Terms, after normalizing the log files by applying adapted and relevant methods (developed in Sections 4.3.1 and 4.3.1), we extract co-occurrences as term candidates. These candidates will be evaluated in the next phase, i.e., Filtering, in order to select the most relevant terms.

4.3.1 Extraction of Terms

The extraction process firstly involves normalization, preprocessing of log files and grammatical tagging of words. Then, the normalized logs are used in the co-occurrence extraction. We detail the different steps of our approach in the following sections.

Preprocessing & Normalization

The heterogeneity of log files can impact the performance of information extraction methods. In order to reduce the data heterogeneity and prepare them to extract terminology, we apply some preprocessing and normalization methods on the logs. The

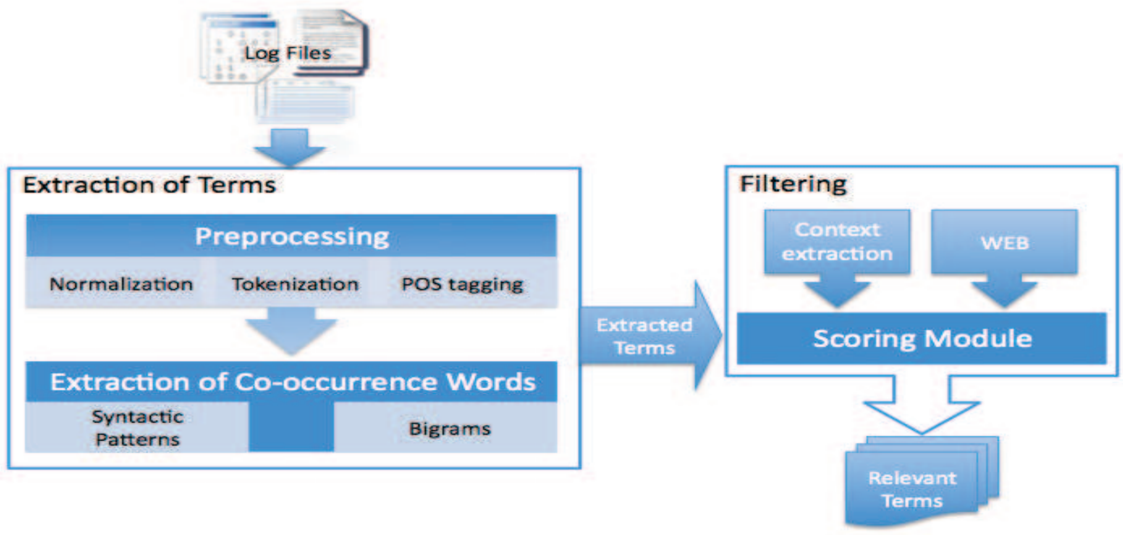


Figure 4.1: Architecture of Exterlog

normalization task mainly concerns data representation formats and log files structure. In order to limit ambiguity in the structure and data representation format, we identify the same punctuations and symbols which are used to represent different notions. According to the log files, we define some special rules that can be applied to distinguish the role of each symbol despite the fact that the symbol can be used for different reasons. For instance, we automatically distinguish lines representing a table header from the lines which separate different parts in a log file. Once the structural role of each symbol is identified, we replace them with a single notation form. There is less ambiguity and less common symbols used for different notions after the normalization process. This normalization streamlines the structure of log files produced by different tools.

Once the normalisation is performed, we tokenize the texts of log files considering the fact that certain words or structures do not have to be tokenized. For example, the technical word “`Circuit4-LED3`" is a single word which should not be tokenized into the two words “`Circuit4`" and “`LED3`". We thus define some tokenization rules which define the border of words in different cases based on their syntax. These rules are defined patterns which describe the syntax of words that should not be tokenized. The definition of rules is carried out manually and according to heuristics and analysing a corpus of log files by help of a domain expert.

Grammatical and Structure Tagging

To identify the role of words in the log files, we use the BRILL rule-based POS tagging method [Brill, 1992]. As described in Section 4.2, existing taggers like BRILL which are trained on general language corpora give inconsistent results on specialized texts like log files. [Amrani et al., 2004] propose a semi-automatic approach for tagging corpora of speciality. They build a new tagger which modifies the base of rules obtained by the BRILL tagger and adapts it to a corpus of speciality.

Since the classic rules of BRILL are not relevant to log files, we have to adapt the BRILL tagger. To give an example, a word beginning with a number is considered as "*cardinal*" by BRILL, while there are many words like `12.1vSo10` in log files that must not be labelled as "*cardinal*". Therefore, in order to take such issues into account, we adapted BRILL to the context of log files by introducing new *contextual* and *lexical* rules. We actually defined about 25 new rules after an in-depth analysis of texts of log files. These new contextual and lexical rules represent grammatical rules existing in log files. They also determine exceptions. For example, we replaced the existing predefined rule in BRILL which says that "all terms beginning with a number are cardinal" with a new one which implies that a term is a cardinal if it does not contain a letter.

Since log file structure could contribute important information for extracting relevant patterns in future work, we preserve their structure during grammatical tagging. For this purpose, we introduce new tags, called "*Document Structure Tags*" representing different structural notions in log files. For example, the tag "\TH" represents table headers, or "\SPL" represents the lines separating different data blocks in log files. Determination and homogenisation of these notions is first accomplished within normalisation process. Then they are identified during the tagging process by the new specific

```
8  | Timing
9  | ——
10 | Warning : Possible timing problems have been detected in this design.
11 |
.. | ...
.. | ...
46 | Clock     Period    Waveform     Attrs     Sources
47 | ————————————————————————————————————————————————-
48 | CLK      12345.67890      {0 2}      {clock}
49 | CLK_tmp      98765.43210      {5 6}      {clock}
```

Figure 4.2: Part of log_B before applying the preprocessing and tagging methods

```
8  | Timing/NN
9  | ——/SPL
10 | Warning/NNP :/: Possible/JJ timing/NN problem/NNS have/VBP been/VBN
   | detected/VBN in/IN this/DT design/NN ./.
11 | —Line—/NEWLINE
.. | ...
.. | ...
46 | Clock/NNP Period/NN Waveform/NNP Attrs/NNP Sources/NNS
47 | ——/TH
48 | CLK/NNP 12345.67890/CD {/( 0/CD 2/CD }/) {/( clock/NN }/)
49 | CLK_tmp/NNM 98765.43210/CD {/( 5/CD 6/CD }/) {/( clock/NN }/)
```

Figure 4.3: Piece of log_B after applying the preprocessing and tagging methods

“contextual rules" defined in BRILL.

We demonstrate the results of normalisation and grammatical tagging on log files by applying our preprocessing method on a piece of log_B presented in Figure 1.5. Figure 4.2 shows the selected part of log_B before preprocessing. Figure 4.3 shows the same part of log_B after applying the preprocessing, normalization, and tagging methods. The grammatical tags /NN, /VB, /JJ, /DT, and /CD correspond respectively to Noun, Verb, Adjective, Determinant, and Cardinal parts-of-speech. The structure tags are coloured in blue Figure 4.3.

Extraction of Co-occurrences

We look for co-occurrences in the log files with two different approaches:

1. Using defined *part-of-speech* syntactic patterns
2. Without using syntactic patterns

The first approach consists of filtering words according to syntactic patterns. The syntactic patterns determine adjacent words having the defined grammatical roles. Syntactic patterns are used by [Daille, 2003] to extract terminology. For complex term identification, [Daille, 2003] defines syntactic structures which are potentially lexicalisable. As argued by [Daille, 2003], base structures of syntactic patterns are not frozen structures and they accept variations. We call the co-occurrences extracted by the first solution, which is based on the syntactic pattern, "POS-candidates". According to the terms found in our context, we do not look for terms consisting of more than two words. Thus, the syntactic patterns that we use to extract POS-candidates in log files are:
"\JJ - \NN" (Adjective-Noun),
"\NN - \NN" (Noun-Noun).

Co-occurrences extracted by the second approach are called "bigrams". A bigram is extracted as a series of any two adjacent relevant words[1]. Bigrams are used in NLP approaches as representative features of a text [meng Tan et al., 2002]. However, the extraction of bigrams does not depend on the grammatical role of words. To extract significant bigrams, we normalize and tokenize the logs to reduce the noise rate. In this method, we do not filter words according to their grammatical roles.

4.3.2 Filtering of Candidates

There are many extracted candidate terms due to the size of log files and the large vocabulary of this domain. However, all extracted terms are not necessarily relevant to the domain. Thus, we need to evaluate and score extracted terms according to their relevance to the context. In order to evaluate extracted terms, we develop and

1. The relevant words, in our context, are all words of the vocabulary of this domain excluding stop words like "have" or "the".

extend our evaluation method proposed in [Saneifar et al., 2011a]. Here we take the determination of context into account as a factor which can influence the evaluation of extracted terms. Thereafter, we present our evaluation function and then how we determine the context of documents from which the terms are extracted.

Web mining ranking

According to the particular features of such data, in spite of the adapted normalization and tagging methods that we have used, some noise exists which result the extraction of irrelevant terms. Moreover, we are focused on a specialized domain where just some terms are really associated with the domain's context. Thus, we evaluate and score the extracted terms according to their relevance to the context. Then we filter the terms having a low score in order to favor the most relevant terms. In order to evaluate the terms, statistical measures are often used in the terminology extraction field (see [Daille, 1996a]). The following are the most widely used.

Mutual Information. One of the most commonly used measures to compute a kind of relationship between words composing what is called a **co-occurrence** is Church's Mutual Information (MI) [Church and Hanks, 1990]. The simplified formula is the following where *nb* designates the number of occurrences of words and pairs of words in a corpus:

$$MI(x,y) = \log_2 \frac{nb(x,y)}{nb(x)nb(y)}$$

Cubic Mutual Information. Cubic Mutual Information is an empirical measure based on MI that enhances the impact of frequent co-occurrences, which is absent in the original MI [Daille, 1994].

$$MI3(x,y) = \log_2 \frac{nb(x,y)^3}{nb(x)nb(y)}$$

Dice's Coefficient. An interesting quality measure is Dice's coefficient [Smadja et al., 1996]. It is defined by the following formula based on the frequency of occurrence of terms.

$$Dice(x,y) = \frac{2 \times nb(x,y)}{nb(x)+nb(y)}$$

This measure is used in several studies related to noun or verb terms extraction in texts [Roche et al., 2004].

These measures are based on the occurrence frequency of terms in the corpus. Scoring terms based on frequencies of terms in the log corpus is not a relevant approach in our context. As we have already explained, techniques based on the *occurrence frequency* of terms in a *corpus* are not relevant to this context as a *representative term* does *not* necessarily have a *high frequency* in log files.

Thus, we score terms according to their occurrence frequency on the Web [2] as a large corpus where the frequency of a term can be representative [Turney, 2001]. However, we obtain bias scores based on the simple count of occurrences of a term on the Web as we are dealing with a specialized domain. Indeed, on the Web, we capture occurrences of terms regardless of the context in which they are seen. That is why we should only consider occurrences of terms on the Web which are located in the EDA context. We therefore use an extension of described measures called $AcroDef$, for which the context and Web resources are essential characteristics to be taken into account (see [Roche and Prince, 2008]). The formulas presented below, define $AcroDef$ measures, based on MI and Cubic MI respectively.

$$AcroDef_{MI}(a^j) = \frac{nb(\bigcap_{i=1}^{n} a_i^j + C)}{\prod_{i=1}^{n} nb(a_i^j + C | a_i^j \notin M_{stop-words})} \quad \text{where } n \geq 2$$

$$AcroDef_{MI3}(a^j) = \frac{nb(\bigcap_{i=1}^{n} a_i^j + C)^3}{\prod_{i=1}^{n} nb(a_i^j + C | a_i^j \notin M_{stop-words})} \quad \text{where } n \geq 2$$

The nb function used in the preceding measures represents the number of Web pages provided by a search engine with a given query. Thus, $nb(a_i^j + C)$ stands for the number of pages (i.e., links) returned by applying the query $a_i^j + C$ to a search engine. This query means all words of the term a^j in addition to those of context C.

2. We define the occurrence frequency of a given term on the Web as the number of pages in which the term is present.

In *AcroDef*, the context "*C*" is represented by a set of significant words. In our case, for example, for a term x^j like "`atpg patterns`" consisting of two words (i.e., $i = 2$), $nb(atpg \bigcap patterns + C)$ is the number of pages returned by giving $\ll$ "`atpg pattern`" AND $C \gg$ as a query to a search engine. Here C is a set of words representing the EDA context. Actually, the objective is to measure how these two words are dependent in the given context. By means of this measure we take the contextual information into account too.

The $AcroDef_{Dice}$ formula [Roche and Prince, 2008] based on Dice's formula is written as follows:

$$AcroDef_{Dice} = \frac{|\{a_i^j + C | a_i^j \notin M_{stop-words}\}_{i \in [1,n]}| \times nb(\bigcap_{i=1}^{n} a_i^j + C)}{\sum_{i=1}^{n} nb(a_i^j + C | a_i^j \notin M_{stop-words})}$$
where $n \geq 2$

The extracted terms are ranked according to their *AcroDef* scores. We favor the most ranked terms by filtering those having the lowest *AcroDef* scores.

The choice of words representing the context impacts the results obtained by *AcroDef*. In [Roche and Prince, 2008], context "*C*" is represented as a set of words (*e.g.* encryption, information, and code to represent the Cryptography context)[3]. The right and exact choice of the domain has a great impact on the evaluation of the results obtained by *AcroDef*. As described, the main motivation of using *AcroDef* is to consider only the occurrence of terms on the Web, which are bound to the studied domain. Working on a specialized domain where each log file corresponds to a more specialized sub-domain, the choice of context requires expertise to obtain the best results. Since human expertise is not often available, we aim at selecting the most relevant words representing the contextual domain in an automatic way. In the next section, we describe how we select words representing the context.

3. In this section, we use simply the term "context" as the set of words representing it.

Context Extraction to Extend Statistical Measures

To specify the words which represent the context of log files, we need to select the most significant words occurring in the log files. We hence use the *tf-idf* scoring function which measures the relevance of words to the domain in which they appear [Salton and Buckley, 1987]. *tf-idf* is based on the hypothesis that a significant word of a domain is frequent in the text of that domain, but less frequent in the text of other different domains.
In a corpus consisting of different documents, the number of times a term occurs in a document is called the *Term Frequency* (*tf*). Thus, we have *tf*, defined as follows:

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

where $n_{i,j}$ is the number of occurrences of the considered term t_i in document d_j. In order to normalize the *tf* value, we use the sum of the number of occurrences of all terms in document d_j ($\sum_k n_{k,j}$).
Inverse Document Frequency (*idf*) corresponds to the number of documents (in the corpus) which contain the given term. We show below how *idf* is calculated:

$$\mathrm{idf_i} = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

$\mid D \mid$ is the total number of documents in the corpus and $|\{d : t_i \in d\}|$ represents the number of documents (d) where the term t_i appears. Finally, the *tf-idf* score is calculated as:

$$\mathrm{(tf\text{-}idf)_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$$

A high *tf-idf* weight value is obtained by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. The *tf-idf* value for a term will always be greater than or equal to zero.

In order to identify the most significant words of the context by *tf-idf*, we build a corpus of documents including reference documents of Integrated Circuit design and also some documents of other different domains like sports and biology. The diversity of domain of documents in the corpus lets us to identify words which are common in most domains (by using *tf-idf*). These words, which have a low *tf-idf* score, are not relevant for representing a particular context, here EDA context.

We have chosen two main methods in order to determine which kind of words are more relevant to represent the context of log files and thus to be scored by *tf-idf*. In the first method, we only extracted all "*nouns*" from the created corpus and scored them by *tf-idf*. In the second method, in order to identify the most relevant words, we scored all words of the corpus which belong to "*nouns*", "*adjectives*", or "*verbs*" parts-of-speech.

Once the selected words of the corpus are scored using the *tf-idf* measure, from the *IC design documents* we choose *n* terms having top scores as representing words of the context.

Moreover, the choice of context words is possible based on the selection of the most frequent words of the domain documents[4]. In this case, the *tf-idf* score is not considered and the only factor to select the most representative terms is the number of occurrences of terms in the domain documents.

In the *Acrodef* calculation, in order to formulate the query which will be used in a search engine, we can use different logical operators (e.g. AND or OR). By using the AND operator, for example, we query pages containing *all* words in "*C*". However, working on a very specialized domain which contains some more specific sub-domains, we do not get the best results by using an "AND" operator for the words of context. Actually, we argue that due to the nature of the Web, pages which are related to a context do not contain all words representing it. Hence, we look for Web pages containing a given term and *two* or *more* words of the context, i.e., we use both operators "OR" and "AND". Note that the default *Acrodef* measure only uses "AND" operator between the context terms.

All of these methods were experimented to choose the best approach of context determination. The results of experiments are presented in Section 4.4.

4. stop-words are filtered.

4.4 Experiments

We evaluate our approach in three main directions:

– Evaluation of both chosen approaches for extraction of Co-occurrences

– Evaluation of *AcroDef* in terms of the ability to classify extracted terms

– Evaluation of term filtering performance

In all experiments, the log corpus is composed of log files generated by different tools and in different conditions. The size of the log corpus is about 950 KB while each log file contains 10000 words in average. We note that we use the same log corpus used in segmentation (cf. Section 2.6) as well as in passage retrieval (cf. Section 3.7) experiments. The obtained terms, are also used in query expansion tests presented in Section 3.7.1 of Chapter 3.

4.4.1 Evaluation of Co-occurrence Extraction Approaches

We tested two different methods in order to extract terminology from logs:

– Extraction of co-occurrences based on syntactic patterns (POS candidates)

– Extraction of co-occurrences based on bigrams of words

In order to analyse the performance of both approaches, we evaluate the terms extracted by each one. At this stage, we prefer an automatic evaluation of candidates (extracted terms) for two reasons: (1) The huge number of candidates, especially those extracted by the second method, make human expertise difficult; (2) Since our goal, at this level, is just to evaluate the performance of each method and not to measure the real precision of our approach. However, in order to accurately measure performance of our approach, a validation by a human expert, is subsequently carried out to complete the automatic validation.

To automatically evaluate the relevance of the extracted terms, we compare the POS-candidates and bigrams with terms extracted from the *reference documents*. Indeed, for each integrated circuits design tool, there are some manual documents, which explain its principles and details. We use these documents as "*reference experts*" in an automatic validation context. In fact, if a candidate term extracted from logs is also seen in the reference documents, we can consider it as being a valid domain term.

Note that, to extract the domain terminology, we have to use log files and not reference documents because there are some terms that do not appear in reference documents according to their nature. Hence, we could use references as a validation tool, but not as the basis of the domain terminology. We remind that a final human validation make it possible to select domain terms which do not exist in reference documents.

Moreover, in order to assess whether the number of occurrences of terms in log files is significant information, we perform a pruning task. We filter the extracted candidate terms based on their frequency of occurrences in the logs. That means that we select candidate terms having an occurrence frequency of at least 2 (i.e., we do not consider terms that have occurred just once in log files). We argue that there is not enough information in the corpus about the terms that have occurred only once in the corpus. Authors of [Roche and Kodratoff, 2006] also discuss the pruning of terms based on their occurrence frequency.

We calculate the precision and recall for the extracted candidate term as shown below:

$$\text{Precision} = \frac{|Candidates \cap Terms\,of\,ref|}{|Candidates|}$$

Table 4.2 shows the precision of POS-candidates and bigrams before and after pruning.

	Level 1		Level 2		Level 3		Level 4		Level 5	
	POS	Bigrams	POS	Bigrams	POS	Bigrams	POS	Bigrams	POS	Bigrams
Before Pruning	67.7	11.3	20.7	6.5	37.8	9.9	40.1	6.5	19.6	5.1
After Pruning	81.1	10.1	18.0	5.0	37.2	5.9	27.3	7.1	37.1	5.5

Table 4.2: Precision of candidate terms before and after pruning based on reference documents and automatic evaluation.

Comparison of candidate terms with the reference terms (see Tab.4.2) shows that the *terminology extraction* based on *syntactic patterns* is quite *relevant* to the *context of log files*. The precision of POS-candidates is indeed *higher* than that of bigrams.

Our experiments show that an effort in normalization and POS tagging tasks is quite useful for extracting relevant terms.

At this experimental level, in order to evaluate the candidate, the precision is the most adapted measure regarding our context. Indeed, this measure gives the general trend of the quality of terms extracted by each method. Note that to calculate a perfectly adapted precision, we have to manually evaluate all terms proposed by Exterlog.

At this stage, we do not calculate the Recall because there is not a set of domain terms to be used as reference. The building of such a set of domain terms from log files which can be used as reference in the Recall calculation requires a manual and complete extraction of the domain terminology by domain experts. Such a task is very expensive.

Note that the pruning of terms based on their occurrence frequency in the log corpus does not significantly improve the results. As we have already explained, in our context, terms are not generally repeated in log files. Therefore, a *representative term* does *not* necessarily have a *high frequency* in the log corpus.

Validation by Experts

In order to validate the “automatic evaluation protocol" using the reference documents, we asked two domain experts to evaluate terms. First, extracted terms are tagged by a domain expert as *relevant* or *not relevant* according to the context and their usefulness in the logs. Then another expert reviewed the tagged terms by the first expert.

We calculated the percentage of terms extracted by Exterlog and validated using reference documents (automatic evaluation protocol), which are also annotated as relevant by experts. The results show that 84% to 98.1% of the terms validated by our automatic evaluation protocol are really relevant terms according to the experts. This interval is due to some terms which are annotated as “no idea" by experts. If we consider the “no idea" terms as irrelevant, 84% of terms validated by our protocol are really relevant according to the experts. If these terms are not taken into account in the calculation, then 98.1% of the terms are really relevant.

As a conclusion to this experiment, extraction of co-occurrences based on syntactic patterns is more relevant to obtain relevant domain terms. The frequency of occurrences of terms in *log files* is not representative information. Hence, the subsequent experiments are carried out for the terms extracted based on syntactic patterns (i.e., POS candidates).

4.4.2 Evaluation of the ability of $AcroDef$ to classify terms

As previously noticed, the extracted terms by Exterlog from log files are so numerous, which complicates validation by domain experts. Thus, we performed the experiments by selecting a sample of extracted terms into our benchmark. Thus, from the logs of every IC design level, we select the 200 most frequent candidate terms. Since there are less than 200 extracted terms for some levels, the taken sample consists of 700 terms overall.

In this experiment, we aim to study the $AcroDef$ ranking function and its ability to give a high score to relevant terms and low score to irrelevant ones. We evaluate the ranking function used to score the terms (i.e., $AcroDef$) using ROC curves (Receiver Operating Curve). A ROC curve allows us to compare the ranking functions (here $AcroDef$) that classify elements of a data-set into both groups, i.e., *positive* and *negative*. It indicates the ability to put the positives before the negatives. In our case, the ROC curve indicates the ability of $AcroDef$ to give a higher score to relevant terms than to irrelevant ones. An effective ranking function should lead to distributions where positives and negatives are well separated. Using ROC curves, we evaluate how much $AcroDef$ is relevant as a measure to distinguish positive and negative terms.

To better analyse the ROC curves, we calculate the AUC (Area Under Curve) which is a synthetic indicator derived from the ROC curve. AUC is the area between the curve and the horizontal axis. If we order individuals at random, the AUC will be equal to 0.5.

Exemple.

We explain, with an example, how ROC curves work. Let L_1 and L_2 be two lists of

terms ranked by two different functions. We indicate each term (*element of list*) by “+" (i.e., relevant term) or “−" (i.e., irrelevant term).
$L_1 = \{(+),(+),(-),(+),(-),(-)\}$,
$L_2 = \{(-),(+),(-),(-),(+),(+)\}$
Since the two lists are ordered with different functions, the terms have different posi-

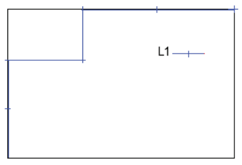


Figure 4.4: ROC curve obtained from L_1

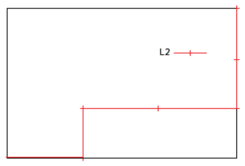


Figure 4.5: ROC curve obtained from L_2

tions. To illustrate the ROC curve, for each + we increase the curve with one unit in the Y axis direction. Also, for each −, the curve is continued with one unit in the X axis direction. As shown in Figures 4.4 and 4.5, the ROC curve corresponding to L_1 is increased on the Y axis more than the ROC curve of L_2. On the other terms, the AUC of L_1 is greater than that of L_2. This shows that the ranking function based on which L_1 is ordered, is more relevant for classifying the positive elements (relevant terms). Moreover, the AUC value of the ROC curve of L_1 is 0.88 when the AUC value of L_2 is 0.22.

As in previous experiments, we asked two domain experts to evaluate the terms ranked by $AcroDef$. The terms were at first tagged by a domain expert as *relevant* or *irrelevant* according to the "IC design domain" and their usefulness in the logs. Then, another expert reviewed the tagged terms by the first expert.

As described in Section 4.3.2, in order to calculate $AcroDef$ values, we use the Google search engine to capture the number of pages containing one given term and *two* or *more* words of context. With one given term like "CPU time" where C_i $i \in \{1-n\}$ are the context words and we take the five top-ranked words (i.e., $n = 5$), the query used in Google search engine is “CPU time" AND C_1 AND (C_2 OR C_3 OR C_4 OR C_5).

To apply *AcroDef*, we determine the context words C, as described in Section 4.3.2, in different ways:

- based on *tf-idf* score:
  - top-ranked words belonging to the POS category "noun"
  - top-ranked words belonging to the POS categories "noun", "adjectives", or "verbs"
- based on the *occurrence frequency* of words (stop-words filtered):
  - most frequent words belonging to the POS category "noun"
  - most frequent words belonging to the POS categories "noun", "adjectives", or "verbs"

In order to determine the best context, we test each method of context determination.

Evaluation of *AcroDef* where the context is determined on the basis of the tf-idf score

Here, we test the *AcroDef* function based on using two different contexts obtained by using the *tf-idf* measure. In the first case, we determine the context by selecting the most ranked words which present "noun" parts-of-speech. In the second case, the context is determined by choosing the most ranked words from a set of words which belong to "noun", "adjective", or "verb" POS categories. In both cases, the words are ranked by a *tf-idf* score.

We calculate the ROC curves according to different filtering thresholds. That is, the number of top-ranked terms by *AcroDef* which are selected as relevant. We consider six thresholds ($m = 200$, $m = 300$, $m = 400, \ldots, m = 700$). With $m = 700$, we actually do not filter any terms as there are 700 terms in our benchmark. We present here the ROC curves obtained with $m = 700$.

Figures 4.6 and 4.7 show ROC curves based on $AcroDef_{MI}$, $AcroDef_{MI3}$, and $AcroDef_{Dice}$ while the context is determined by using *tf-idf* and with $m = 700$. Tables 4.3 and 4.4 show AUC according to the ROC curves based on $AcroDef_{MI}$, $AcroDef_{MI3}$, and $AcroDef_{Dice}$, while the context is determined by *tf-idf*. As described above, the parameter m is the filtering threshold. With $m = 500$, for example, we take the 500

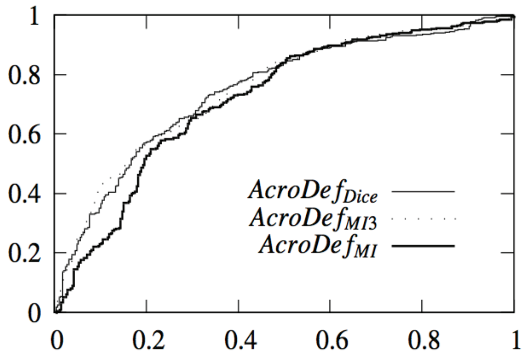


Figure 4.6: ROC curves based on three types of *AcroDef* while the context contains the most ranked “nouns" (using *tf-idf* score and $m = 700$)

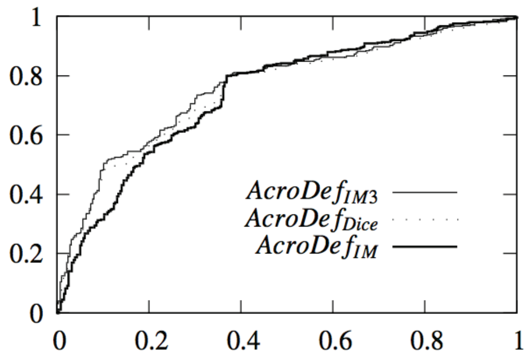


Figure 4.7: ROC curves based on three types of *AcroDef* while the context contains the most ranked words (nouns, adjectives, verbs) by using *tf-idf* score and $m = 700$

m	AUC_{MI}	AUC_{MI3}	AUC_{Dice}
200	0.50	0.50	**0.58**
300	0.48	**0.64**	0.60
400	0.58	**0.66**	0.63
500	0.60	**0.68**	0.67
600	0.67	**0.72**	**0.72**
700	0.71	**0.75**	0.74

Table 4.3: AUC obtained at each filtering level based on the *AcroDef* while the context contains just the most ranked nouns (using the *tf-idf* score)

m	AUC_{MI}	AUC_{MI3}	AUC_{Dice}
200	0.53	**0.60**	0.59
300	0.61	**0.70**	0.66
400	0.62	**0.71**	0.68
500	0.66	**0.74**	0.71
600	0.72	**0.75**	**0.75**
700	0.74	**0.77**	0.76

Table 4.4: AUC obtained at each filtering level based on the *AcroDef* while the context contains the most ranked words (nouns, adjectives, verbs) by using the *tf-idf* score

top-ranked terms.

According to the AUC values, for example, when we use $AcroDef_{MI3}$ and *tf-idf* measures to determine the context, with $m = 500$, if the context is determined by choosing the representative word belonging to *noun*, *adjective*, or *verb* POS categories, it is 74% likely that relevant terms have a higher *AcroDef* score than irrelevant terms. In the same conditions, if the context is represented just by “*nouns*", in 68% of cases relevant terms have a higher *AcroDef* score than irrelevant ones.

According to the results, we see that while the context is determined by words which belong to noun, adjective, or verb POS categories, we have more relevant *AcroDef* functions. This means that this method of context determination is more relevant than others that use the words belonging just to the “noun" POS category.

Evaluation of *AcroDef* while the context is obtained based on the words occurrence frequency

In this section, we have focused on the study of the use of other methods to determine the context. In the last section, context words were scored by the *tf-idf* measure. But here we choose the most frequent words to represent the context. So, the only factor is the number of occurrences of words in domain documents. As described before, we build two different contexts. The first one contains the most frequent words belonging to the “noun" POS category. The second context contains the most frequent words belonging to the “noun", “adjective", or “verb" POS categories.

We also calculate ROC curves according to different filtering thresholds. Like previous ROC curves, we present here the curves obtained with $m = 700$.

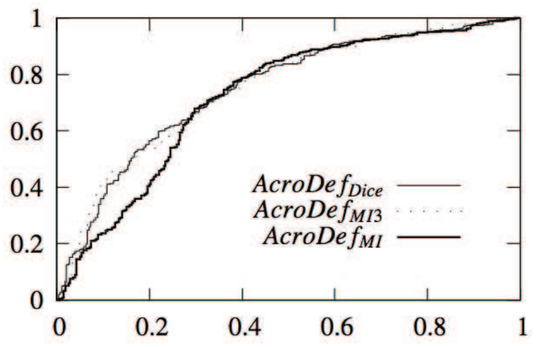


Figure 4.8: ROC curves based on three types of *AcroDef* while the context contains the most frequent nouns and $m = 700$

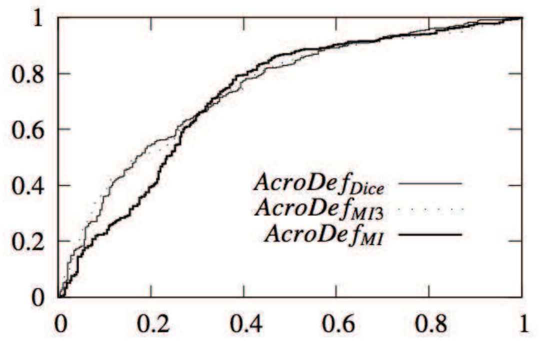


Figure 4.9: ROC curves based on three types of *AcroDef* while the context contains the most frequent words (nouns, adjectives, verbs) and $m = 700$

Figures 4.8 and 4.9 show ROC curves obtained by $AcroDef_{MI}$, $AcroDef_{MI3}$, and $AcroDef_{Dice}$ while the context is determined as described below. Tables 4.5 and

4.6 show AUC corresponding to ROC curves based on $AcroDef_{MI}$, $AcroDef_{MI3}$, and $AcroDef_{Dice}$ while the context is determined by selecting the most frequent words.

m	AUC_{MI}	AUC_{MI3}	AUC_{Dice}
200	**0.57**	0.50	0.48
300	0.51	**0.65**	0.59
400	0.52	**0.64**	**0.64**
500	0.58	**0.67**	**0.67**
600	0.68	0.70	**0.71**
700	0.72	**0.74**	**0.74**

Table 4.5: AUC obtained at each filtering level based on *AcroDef* while the context contains the most *frequent* nouns

m	AUC_{MI}	AUC_{MI3}	AUC_{Dice}
200	**0.56**	0.55	0.53
300	0.50	**0.66**	0.62
400	0.51	0.62	**0.63**
500	0.57	**0.66**	**0.66**
600	0.68	**0.72**	0.70
700	0.72	**0.74**	**0.74**

Table 4.6: AUC obtained at each filtering level based on *AcroDef* while the context contains the most *frequent* words (nouns, adjectives, verbs)

When the context is determined based on the *occurrence frequency* of words and we are using $AcroDef_{MI3}$, according to the AUC results, if the context is represented by words belonging to *noun*, *adjective*, and *verb* POS categories, it is 66% likely that relevant terms have a higher *AcroDef* score than irrelevant ones (when $m = 500$). While, according to our previous experiment (cf. Tab. 4.4), in the same conditions, if the context is determined by using *tf-idf*, we have an AUC of 74%.

To conclude, according to the results, the best method to choose the context is to rank words of documents by *tf-idf* measure and select the most ranked words which belong to noun, adjective, or verb POS categories. Moreover, *AcroDef* calculated based on $MI3$ is more relevant than both other types of *AcroDef*. In our benchmark, in 77% of cases, by using $AcroDef_{MI3}$, a relevant term has a higher score than an irrelevant one.

Finally, in the following section, we evaluate the performance of our terminology extraction in order to find the best filtering threshold (i.e., value of m).

4.4.3 Performance of term filtering

In this experiment, we use $AcroDef_{MI3}$ and the context contains words belonging to *noun*, *adjective*, and *verb* POS categories (based on the tf-idf measure). These conditions are chosen according to the results of the previous experiments. Here we aim at calculating the precision and recall of our approach in terms of relevant term extraction. We focused on determination of the best filtering threshold for our approach.

We ranked terms based on their $AcroDef$ score. Then we filter the terms by selecting the top-ranked ones. Terms having low score are filtered (i.e., pruned). The terms are evaluated by two domain experts as relevant or irrelevant. The precision is calculated as a percentage of remaining terms (*after prunning based on $AcroDef$ scores*) which are tagged as “relevant" by experts.

$$Precision = \frac{|Terms_{relevant} \cap Terms_{remained}|}{|Terms_{remained}|}$$

$Terms_{relevant}$ = terms validated by experts
$Terms_{remained}$ = terms remaining after filtering

We calculate the recall as the percent of all relevant terms (*in benchmark* scale) which remain after filtering.

$$Recall = \frac{|Terms_{relevant} \cap Terms_{remained}|}{|Terms_{relevant}|}$$

$Terms_{relevant}$ = terms validated by experts in the **benchmark**
$Terms_{remained}$ = terms remaining after filtering

We have also calculated the F-score as the harmonic average of precision and recall.

$$F - score = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

To calculate the performance of our approach at different filtering level and also to find the best filtering threshold, we calculated the precision, recall, and F-score while we consider the different filtering thresholds. We filter terms by selecting the *m* top-ranked terms (ranked by the $AcroDef$ score).

m	Precision	Recall	F-score
200	86 %	41 %	56 %
300	79 %	57 %	67 %
400	76 %	74 %	75 %
500	72 %	87 %	**79** %
600	66 %	95 %	78 %
700	59 %	100 %	74 %

Table 4.7: Precision, Recall, and F-score of terms in each level m of filtering

Table 4.7 shows the filtering results with different m values. For instance, with $m = 300$, we take the 300 top ranked terms. With $m = 700$, we do not actually filter any terms, thus, the recall is equal to 100%. The results show that through our filtering approach, we favor more relevant terms and emphasize the precision. According to the results, by using $AcroDef_{MI3}$ as the ranking function while the context is determined by *tf-idf*, the F-score of our approach is 79% (Precision=72% & Recall=87%) if we take the 500 most ranked terms as relevant. To obtain better precision, we have to decrease the filtering threshold, but the recall will decrease.

4.5 Discussions

We had demonstrated in Chapter 3 that integrating the terminological knowledge of the domain into process of query expansion and passage retrieval can significantly improve the results. Thus, in this Chapter, we presented why and how to extract the terminology of log files. In fact, the specificities of log files make the classical method of terminology extraction inefficient.

In Section 4.1, We first introduced the terminology notion and terminology extraction. We also motivated our work in this domain. Section 4.2, was devoted to study the existing methods in this domain. In this section, we presented the main directions: Use of syntactic patterns, statistic measures, lexical dependency, and machine learning based approaches. Most of the current work are tested on classical texts written in a natural language which are consistently structured and comply with NL grammar.

We argued that this methods have to be adapted to ensure that they are relevant in the context of log files. For example, we discussed the difficulties and limitations for applying grammatical tagging and hence using the syntactic patterns on such textual data. In addition, we explained that the use of statistic methods, as they are, to evaluate and validate the candidate terms rises some difficulties in the context of log files.

In Section 4.3, we developed our approach of terminology extraction from log files, called Exterlog. We also explained how to pre-process and normalize the log files in order to prepare them to apply NLP methods. In addition, we adapted a POS tagging method to the characteristics of log files. The extraction of terms was performed using defined syntactic patterns. We finally proposed an extended Web and context based statistical measure and a term evaluation protocol by considering the specificities of log files. Using these adapted methods and the proposed evaluation protocol, we overcame the difficulties seen in the extraction of terms in log file corpus.

We presented some experimental results in Section 4.4. The experiments showed that our approach for terminology extraction from log files, Exterlog, can achieve a F-score equal to 0.79 after filtering terms once they are evaluated by our term validation protocol.

Th extracted and validated terms are used in passage retrieval and query expansion. In all tests performed in previous chapter about the query expansion, we used the extracted terms to obtain more relevant expansion terms. The use of terminological knowledge in these tests is explicitly discussed in Section 3.7 of Chapter 3. The extraction of the domain terminology is also considered as the first step in the creation of the domain ontology and semantic resources. We discuss this point in next chapter, under the future work section.

Chapter

5

Industrial Integration of the System

In theory, there is no difference between theory and practice. But, in practice, there is.

Jan L.A. van de Snepscheut

Preamble

This chapter is devoted to the industrial application of our approach. In fact, our contribution is a solution to facilitate the quality maintenance in the EDA domain. It hence needs to be integrated to a quality monitoring software, called VIP Lane, which is designed by Satin Technologies. We explain in this chapter how our approach is integrated into the VIP Lane product.

Contents

5.1 Introduction

This research project is defined within an industrial-academic collaboration between the Satin Technologies[1] company and the LIRMM[2] research laboratory. This means that we work on a real industrial issue and process the real world data (i.e., log files) provided by Satin Technologies. The contributions of this thesis are hence required to be implemented and integrated to the current Satin Technologies commercial software called VIP Lane.

VIP Lane is an enterprise software solution for design quality monitoring. Once deployed, VIP Lane is a web server application that is integrated with and complements the electronic design automation (EDA) and product life-cycle management tools that typically constitute semiconductor design flows. The objective is to make it enable to turn design practices (e.g., for IP blocks or embedded systems) into a robust and reliable set of quality criteria and metrics. By means of VIP Lane, one can monitor the design quality in a factual and cost effective way. He can verify how his work complies with corporate quality rules and metrics without additional engineering effort.

In VIP Lane, the quality criteria are defined in the form of questions (called quality checks) and log files generated by EDA design tools are supplied as the main data sources. In order to answer quality check questions, VIP Lane currently provides solutions which need an amount of manual work to configure the methods which look for the requested data. These methods are called sensors in VIP Lane. Our solution to locate the requested information in the mass of the heterogeneous and evolving log files takes place at the sensor levels. By means of our approach we help experts to find information in a smarter way and significantly decreases the manual preparing work.

Locating the requested information by means of our passage retrieval approach makes it possible to define really less sophisticated answer extraction patterns. The current used patterns need to be complex and sophisticated in order to ensure the extraction of the exact information and not any other syntactically similar one. By means of our approach, we eliminate any other similar information in passage retrieval and

1. www.satin-tech.com
2. www.lirmm.fr

isolate the requested information by locating the requested information via our approach.

In the rest of this chapter, we develop the integration of our approach into VIP Lane. Although our solution is currently integrated into VIP Lane following the software Engineering process, we avoid to detail the design and the specification steps as it is out of this dissertation's scope. Note that our approach has successfully passed the first tests and release process.

5.2 Integration into VIP Lane

First we introduce all the components and steps of our approach and the way they are executed. For this purpose, Figure 5.1 presents the global activity diagram of our approach. Activity diagrams are graphical representations of workflows of stepwise activities and actions. In the Unified Modeling Language, activity diagrams are used to describe the step-by-step workflows of components in a system.

For a given question (sensor), the system takes two log files. One for training purpose and the other one (called test log file) is the real data source wherein we look for answer. In the real usage of VIP Lane, although test log file has the same type as the training log file (i.e., they are both generated in the same conditions and level of design and thus mostly contain the same information), it is usually an evolved version of the training file or it is generated by a different design tool.

Preprocessing

Before performing any training or information extraction on log files, they need to be preprocessed. Preprocessing consists of three steps: (1) normalization, (2) grammatical tagging, and (3) segmentation. The two first steps are explained in Section 4.3.1 of Chapter 4. Our segmentation method is also developed in Chapter 2.

Since the second step (grammatical tagging) is not absolutely useful and was only necessary to extract the domain terminology, we do not integrate it into VIP Lane. That mainly helps to obtain a better performance at processing of huge log files.

Although the normalization and segmentation are two independent steps, they are

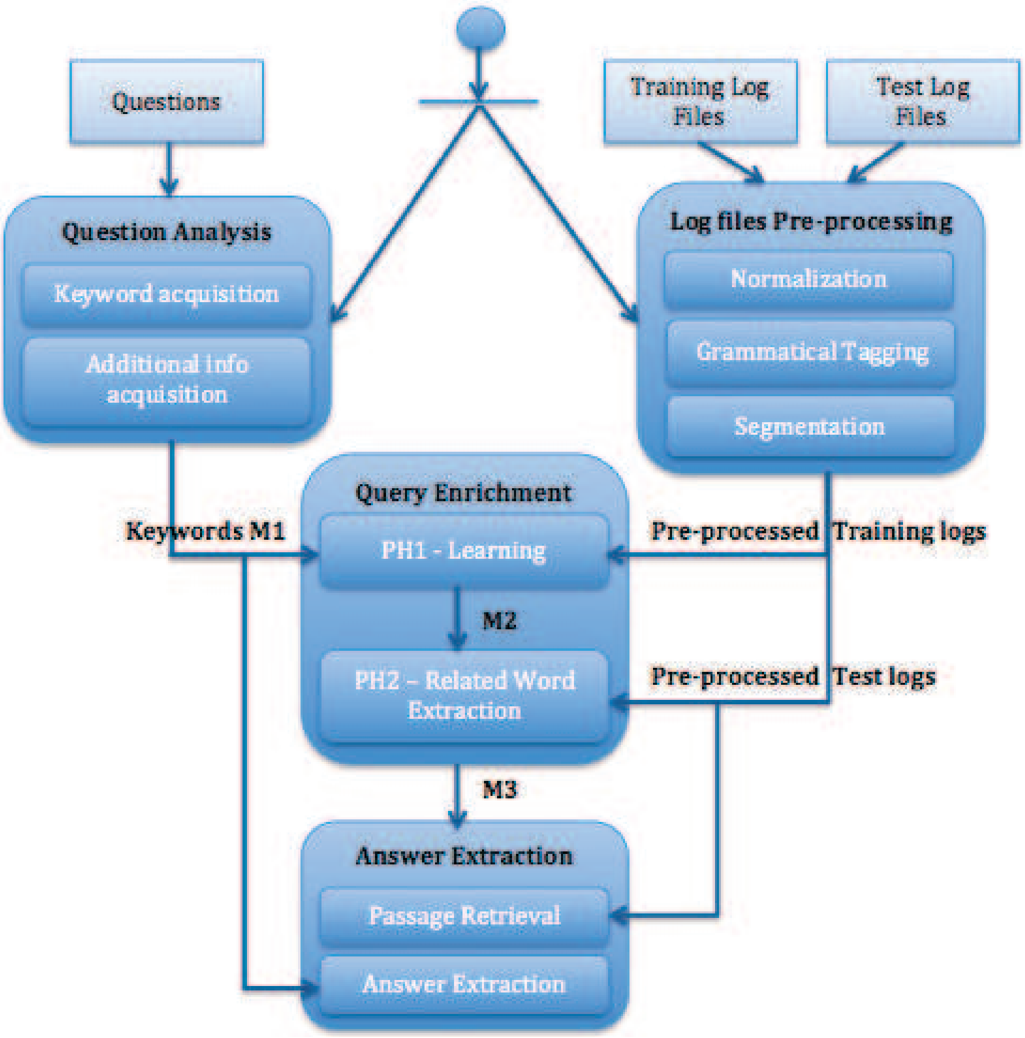


Figure 5.1: Workflows of steps in our approach to locate answers in log files.

implemented within an unique method. This means that they are both performed simultaneously in a parallel way which increases the performance of our approach in terms of execution time.

The output of preprocessing is a new normalized log file which contains the annotations showing the start of each segment in the log file.

Question Analysis

The goal is to obtain some relevant information from questions. This information are subsequently used in training and answer extraction. At this level, we only consider question keywords as the supplied information. Question keywords can be supplied manually by the expert who defines questions or be obtained from questions by extracting their main terms based on pre-defined syntactic patterns. For example, for the question "capture the different clock domain periods", we have "clock", "domain", and "period" as the initial keywords. Other words are not enough informative (i.e., they are too general words) or are the stop-words. The output of this phase is a list of keywords which are called "initial keywords", shown as $M1$ in Figure 5.1.

Query Enrichment

As explained in Chapter 3, the objective of query enrichment is to make it possible to look for a requested information in different log files without any further change in methods or in the definition of questions.

Enrichment consists of two phases: (1) training (see Section 3.4.1) (2) extraction of terms related to answer (see Section 3.5). Training of a query needs the initial keywords ($M1$) and the preprocessed training log file. The output is a list of keywords which will be integrated into the initial keywords to obtain a new list of keywords called $M2$ in Figure 5.1. We note that the training step is only performed once for a question even though the test log file can change.

In the second phase of query enrichment, we use directly the test log files. This means that the answer related terms are seek in test log files. That is why it is required to perform this phase at every change of test log files. The result is again a new list of keywords. By integrating them into $M2$, we obtain the $M3$ keyword list.

Answer Extraction

Inputs are the test log file and the enriched query (i.e., $M3$). At the first step, we look for passages which likely contain the answer. Creation of final queries and passage retrieval using Lucene[3] are developed in Section 3.4.2. At this level, the number of extracted passages are configurable in VIP Lane. At the second step, we seek for the answer in the extracted passages. In the current version, we capture the answer by means of defined patterns. These patterns are provided in the sensor properties by the domain expert. As previously explained, thanks to our approach, it is not required to define the complex, sophisticated, and time consuming answer extraction patterns. Beside simplifying the answer extraction patterns, locating answers by means of our approach increase considerably the performance of the system.

5.3 Demonstration

In VIP Lane, we consider two independent phases in order to define a question and answer it: (1) creation of sensors and (2) execution of sensors. Sensor creation consists of defining a question by means of a sensor and preparing it for execution. These phase requires a given amount of manual work in the classic version of VIP Lane (i.e., without the use of our approach). Execution phase consists of actions which take place in order to look for the information requested in the sensor definition.

In this section, we demonstrate how to create and execute a sensor using our approach. By this demonstration, we explain which components and steps of our approach are used in each phase (i.e., creation and execution of sensors).

5.3.1 Sensor Creation

We first present in Figure 5.2 the workflows of all steps of our approach included in the creation of a sensor. This diagram shows the activities in VIP Lane to create a sensor using our approach.

User first creates a sensor by providing some information regarding the question including an abstract description and eventually the initial keywords. Then sensor creation process continues by supplying a test log file to the sensor as well as another log file to

3. http://lucene.apache.org/

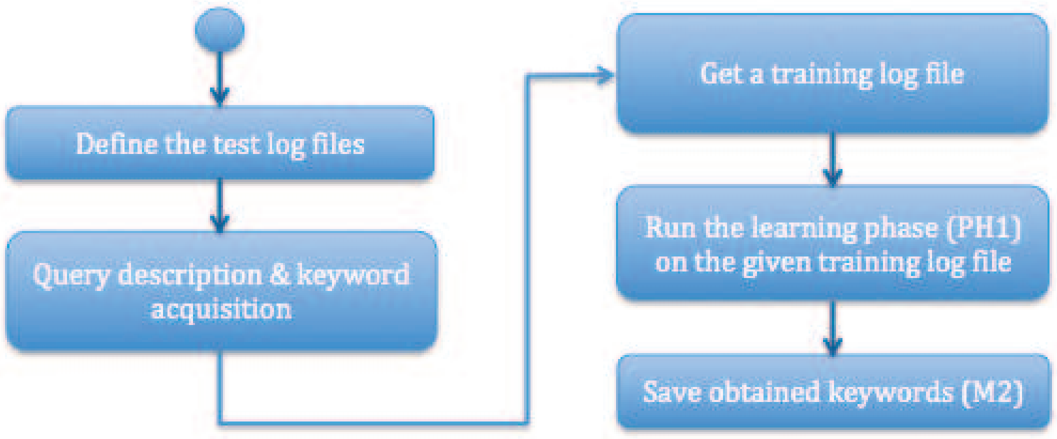

Define the test log files
Query description & keyword acquisition
Get a training log file
Run the learning phase (PH1) on the given training log file
Save obtained keywords (M2)

Figure 5.2: Steps to create a sensor in VIP Lane

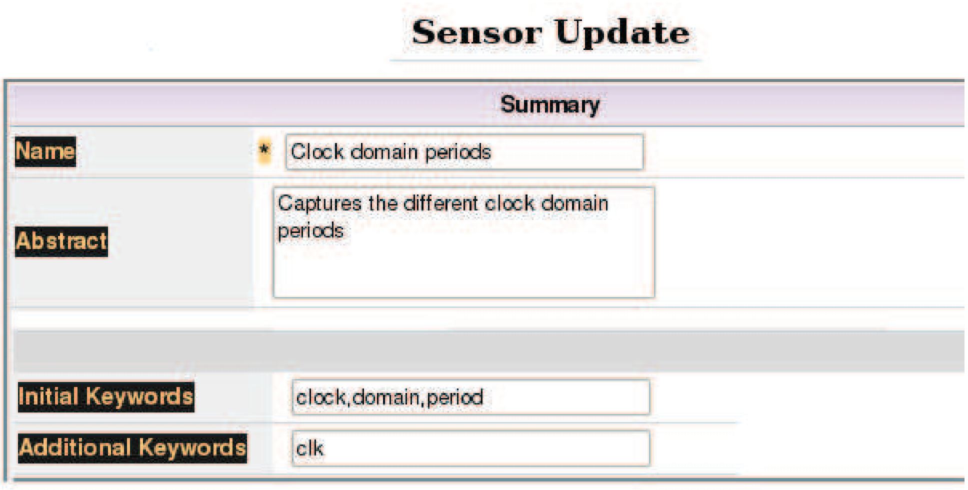
be used in the training phase (i.e., training log file).

Figures 5.3 shows the two parts of the VIP Lane interface used to create a sensor. Highlighted fields in Figure 5.3 are devoted to information regarding the question as well as the initial keywords. User has the possibility to provide additional keywords beside those found in the text of question (i.e., initial keywords). Fields to use in order to provide log files are also demonstrated in Figure 5.4. Data file field in the interface corresponds to the test log file.

Once the entries are supplied, the user should perform the training process to accomplish the creation of the sensor. Training process starts using the initial keywords and the training log file. If the training log file needs to be preprocessed (i.e., it is not previously performed), it will take place before the training.

By launching the training, the system identifies the context of the question as described in Section 3.4.1. At this stage the system shows the passages in the training log file which present at best the context of question (see Figure 5.5). Then the user selects the passage representing the context of the question at best among those proposed by the system. We note that, it would be possible to select several passages at this level. This can happens in general domain applications as explained in Section 3.6.

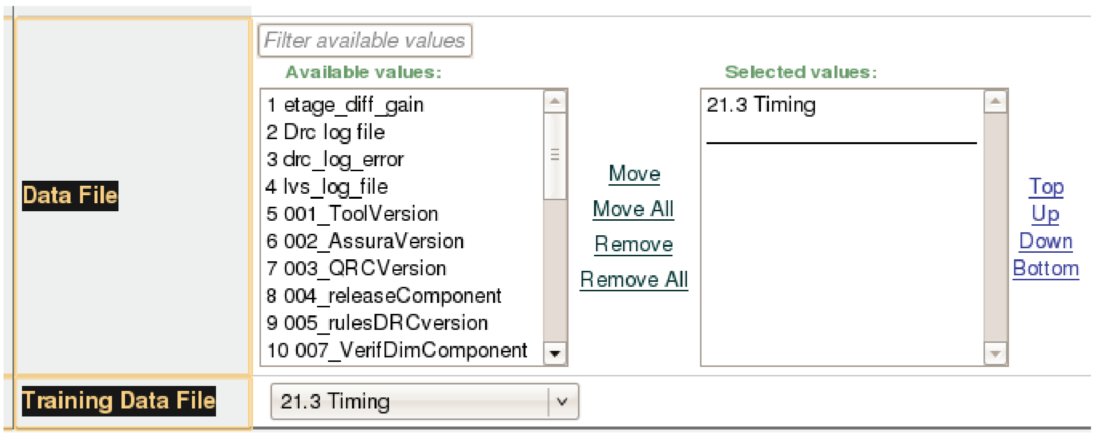
The keywords learnt at the end of the learning phase are saved in the sensor properties.



Sensor Update

Summary	
Name	* Clock domain periods
Abstract	Captures the different clock domain periods
Initial Keywords	clock,domain,period
Additional Keywords	clk

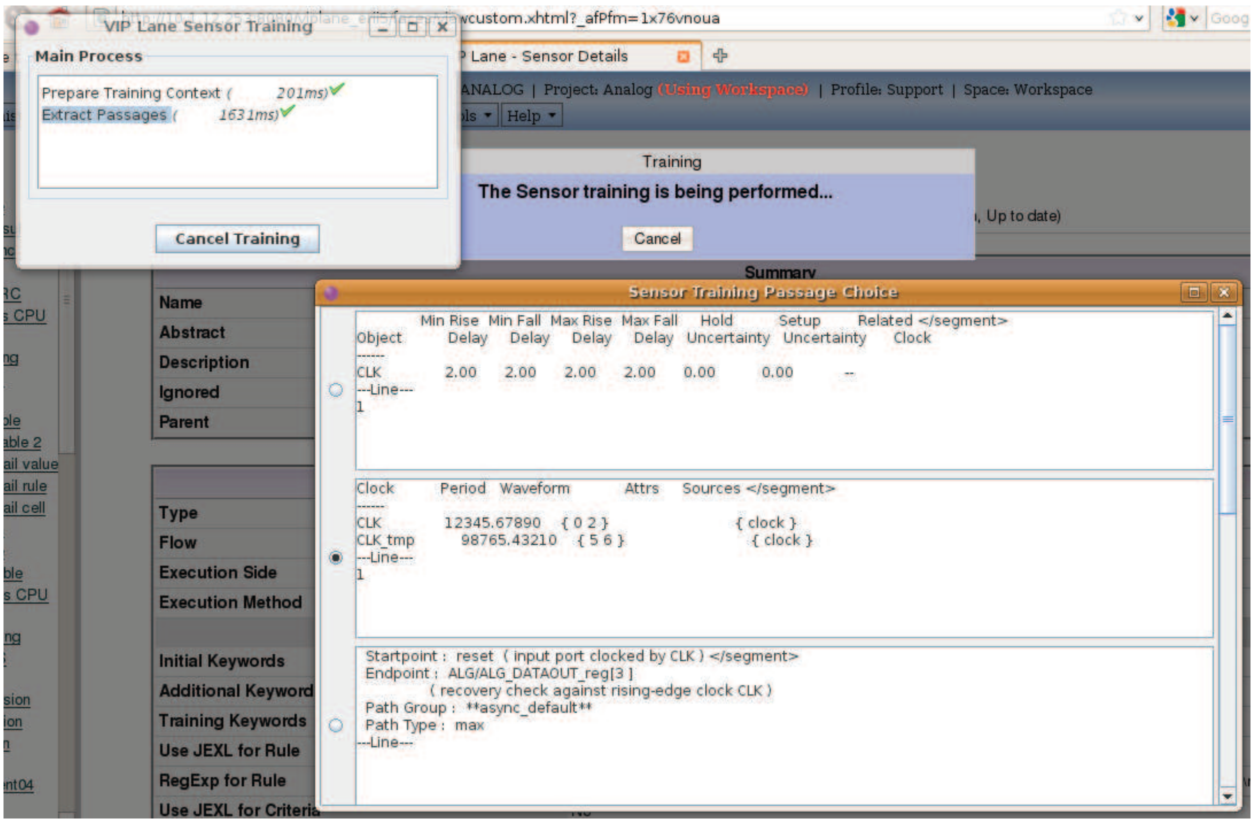
Figure 5.3: VIP Lane interface for the question definition



Data File	Filter available values Available values: 1 etage_diff_gain, 2 Drc log file, 3 drc_log_error, 4 lvs_log_file, 5 001_ToolVersion, 6 002_AssuraVersion, 7 003_QRCVersion, 8 004_releaseComponent, 9 005_rulesDRCversion, 10 007_VerifDimComponent Selected values: 21.3 Timing
Training Data File	21.3 Timing

Figure 5.4: VIP Lane interface for log file reference

At this stage, the sensor is prepared for execution. Figure 5.6 shows the keywords learned after training which are imported to the sensor properties.


VIP Lane Sensor Training

Main Process

Prepare Training Context (201ms)✓
Extract Passages (1631ms)✓

Cancel Training

Training

The Sensor training is being performed...

Cancel

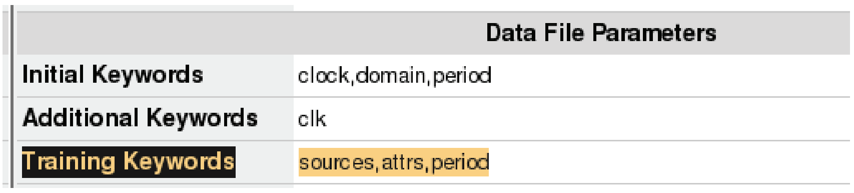
Sensor Training Passage Choice

```
        Min Rise  Min Fall  Max Rise  Max Fall  Hold         Setup        Related </segment>
Object  Delay     Delay     Delay     Delay     Uncertainty  Uncertainty  Clock
------
CLK     2.00      2.00      2.00      2.00      0.00         0.00         --
---Line---
1
```

```
Clock     Period      Waveform   Attrs   Sources </segment>
------
CLK       12345.67890  { 0 2 }            { clock }
CLK_tmp   98765.43210  { 5 6 }            { clock }
---Line---
1
```

```
Startpoint :  reset  ( input port clocked by CLK ) </segment>
Endpoint :  ALG/ALG_DATAOUT_reg[3 ]
         ( recovery check against rising-edge clock CLK )
Path Group :  **async_default**
Path Type :  max
---Line---
```

Figure 5.5: Training process in VIP Lane



	Data File Parameters
Initial Keywords	clock,domain,period
Additional Keywords	clk
Training Keywords	sources,attrs,period

Figure 5.6: Training keywords integrated to the sensor properties

5.3.2 Sensor Execution

We first introduce the steps to execute a sensor using our approach. Figure 5.7 presents all activities included in the execution of a sensor.

When the sensor execution is launched, it first verifies whether the test log file is preprocessed or not. Once having a preprocessed test log file, the second phase of

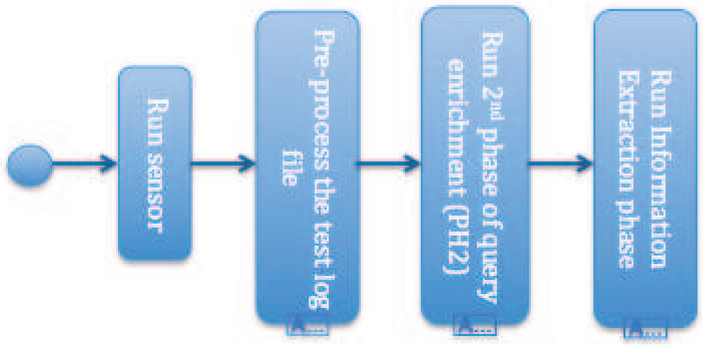
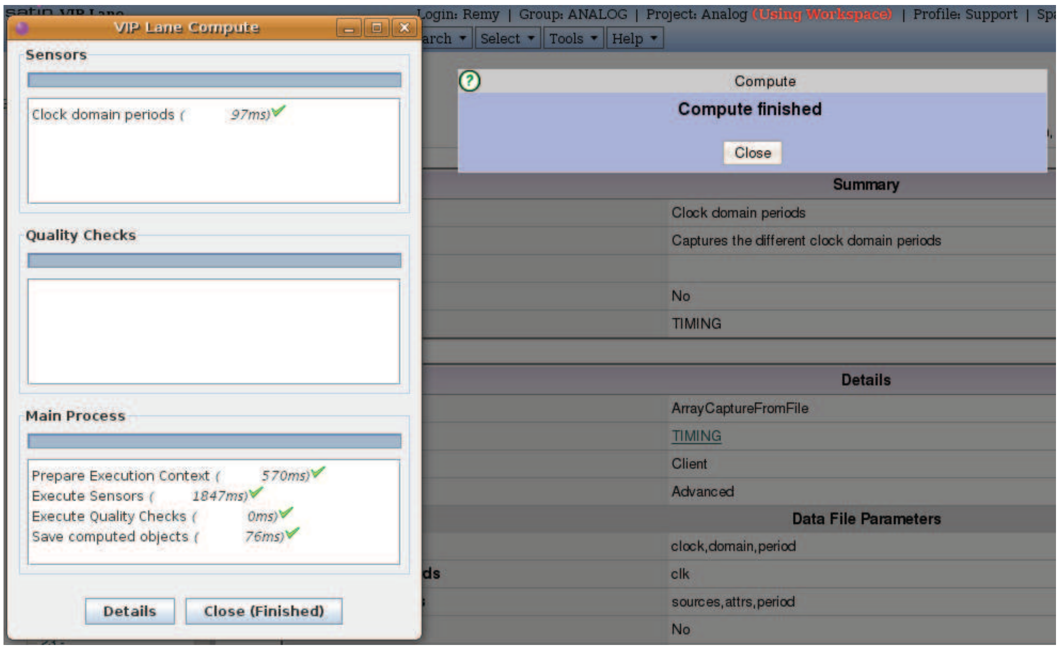

Run sensor
Pre-process the test log file
Run 2nd phase of query enrichment (PH2)
Run Information Extraction phase

Figure 5.7: Training keywords integrated to the sensor properties


VIP Lane Compute
Login: Remy | Group: ANALOG | Project: Analog (Using Workspace) | Profile: Support
arch
Select
Tools
Help
Details
Close (Finished)
Close

Sensors

Clock domain periods (97ms)✓

Quality Checks

Main Process

Prepare Execution Context (570ms)✓
Execute Sensors (1847ms)✓
Execute Quality Checks (0ms)✓
Save computed objects (76ms)✓

Compute

Compute finished

Summary

Clock domain periods
Captures the different clock domain periods
No
TIMING

Details

ArrayCaptureFromFile
TIMING
Client
Advanced

Data File Parameters

clock,domain,period
clk
sources,attrs,period
No

Figure 5.8: Sensor execution process

the query enrichment (answer related words extraction) starts. This phase of query expansion does not need any human intervention or external knowledge. After obtaining the expanded query, the passage retrieval phase automatically starts. Figure 5.8 shows a sensor during the execution process. The answer extraction just starts after the passage extraction.

Chapter 6

Conclusion

One pound of learning requires ten pounds of common sense to apply it.

Persian Proverb.

Preamble

In this chapter we first summarize this thesis and then present some future directions regarding the presented contributions.

Contents

6.1 Summary

In this thesis, we investigated the extraction of information in a kind of complex and evolving textual data, i.e., log files generated by EDA design tools. These log files contain essential information which can be used in the EDA quality verification process. Our objective was to propose a solution which makes it possible to locate a requested information in log files. For this purpose, we had to consider the specificities of log files, presented in Chapter 1, which make the classic methods of information retrieval and natural language processing inefficient.

We first discussed, in Chapter 2, the structure of log files and the problem of segmenting them into relevant text chunks, called passages. In fact, as described, segmenting textual data into passages is an essential task in order to locate a requested information. We described why considering simply a line or a sentence as a passage is not relevant in the context of log files. Then we presented the current work about text segmentation including the main approaches, i.e., semantic, window-based and discourse passages. As shown in Chapter 2, the semantic segmentation methods principally rely on the use of word occurrence frequency, lexical co-occurrences, or lexico-semantic relations to identify subject changes in documents. We described that a change of co-occurrences or a change of term repetition do not necessarily result in topic change in log files. By means of some real examples, we showed that the methods based on term iteration or term occurrences are not always relevant. We therefore demonstrated that window-based methods can result into information lost. We hence investigated the relevance of methods which are called discourse passages. The main idea behind these methods is to recognize the logical units of documents. We proposed an approaches to characterise and identify these logical units.

In our approach we created a set of features where each one presents a syntactic characteristic of logical units. To build the set of features, we have proposed two methods: semi-automatic approach (via heuristics of an expert), and automatic approach (via extraction of generalized vs-grams). The results show that *the generalized vs-grams*, introduced in this thesis as a new kind of grams, can be used for modelling the complex logical units or the visual structure of documents.

Using a training set based on the obtained features and a supervised classification method, we built a classification model which made it possible to distinguish the logical units in the log files. These results have confirmed the experts to use our segmentation protocol in their industrial usages.

In Chapter 3, we investigate how to enhance the relevancy of queries during a passage retrieval in log files. Passage retrieval is aiming at retrieving relevant passages in documents which contain answers to a questions. In simple words, the objective is to locate a requested information in documents. In this thesis, we demonstrated that the specificities of log files can impact the performance of passage retrieval. In most of cases, a given query is not relevantly formulated to be used for all types of log files. Vocabulary heterogeneity and the fact that the query keywords do not necessarily exist in all types of log files make passage retrieval difficult. That is why we proposed a query expansion approach to overcome the difficulties of this domain, notably mismatch vocabularies.

Our proposed query expansion approach relies on two novel relevance feedback steps. The first one, being a kind of explicit relevance feedback system, the feedbacks are obtained by identifying the *context of queries* within a supervised learning process. In order to identify and learn the context of queries, we proposed a new method based on the "lexical world" notion. We used the determined context of queries as relevant documents wherein we looked for expansion terms.

The second step was a new kind of pseudo relevance feedback. Contrary to most pseudo relevance feedback methods considering the initial top-ranked documents as relevant, we introduced a novel method which is based on a new term weighting function, called TRQ (Term Relatedness to Queries), which gives a score to terms of corpus according to their *relatedness* to the *query*.

We presented TRQ measure as an original term weighting function which aims at giving a high score to terms of the corpus which have a significant probability of existing in the relevant passages. We remind here that in the second step of query expansion there is no information about the relevance of the passages as we directly use the test corpus. Expansion terms were finally selected according to their TRQ scores.

Despite of the log file characteristics, our approach made it possible to adapt an initial query to all types of corresponding log files regardless of their vocabulary. According to the results, by using our query expansion protocol, we obtained satisfactory results.

We also evaluated the application of our approach in general domains. For this purpose, we used the TREC'04 evaluation campaign documents. In this part, we studied the difference between the application of our approach in specific and general domains. The results showed that our approach, after a few adaptations, is also relevant in general domains. The main adaptations were performed according to the fact that there are several occurrences of an answer in general documents. This points needed to be considered in the calculation of TRQ measure and the selection of relevant passages in the training phase. We obtained satisfactory results after enriching the initial queries by using our approach.

In Chapter 4, we studied how to acquire the EDA domain terminological knowledge from the log files. Indeed, we discussed in Chapter 3 that the use of terminological knowledge can improve the performance of query enrichment. Use of multi-word terms in query expansion has improved in many cases the relevance of obtained queries. Although the usefulness of the domain terminology in query expansion and passage retrieval was demonstrated, the acquisition of the domain terminological knowledge was not a trivial task in the case of log files. We discussed the challenges which exist in terminology extraction in log files. We argued why the classic methods of NLP are not by default relevant in the case of log files. We subsequently proposed our approach, called Exterlog, to extract the terminology of log files. Exterlog is based on the extraction of co-occurrences. In order to extract the terminology of log files, we also applied a specific preprocessing, normalization, and grammatical tagging within Exterlog approach.

We also discussed the terminology validation methods in the context of log files where the occurrence frequency of terms is not significant. We proposed a term validation protocol which aims to reduce the noise ratio in extracted terms and favor more relevant terms of this domain. For this purpose, we scored the initially extracted terms using a Web mining approach. This one is based on statistical measures which are based on Web and contextual information. Our term validation protocol enables to select the most relevant terms of the domain. The experiments showed that our approach for terminology extraction from log files, Exterlog, achieved a high F-score after filtering terms.

6.2 Future Work

We conclude this thesis by presenting the future work in this domain. These perspectives include the following directions.

6.2.1 Extension of Retrieval Methods and Measures

In query expansion, there are few parameters which can eventually have an influence on the expansion term selection. For example, we can study whether considering the distance of terms from the query keywords is a relevant factor in the context of log files. In fact, the distance of terms from the query keywords is usually an informative parameter in general domain documents. However, in log files it needs more study to define exactly if it is informative or not.

For this purpose, the first point that should be defined is the notion of distance between two terms in log files. We should take the cases like tables into account wherein a distance based on the number of terms is not significant. Moreover, the visual distance between two terms in a log file can differ from their distance in the same context in another log file. A solution can be to select the expansion terms in a part of text which have the highest keyword density. Based on the number of keywords in a selected passage, we can identify several fragment starting with a query keyword and ending with another one. Then, we should calculate the density of keywords in each fragment to select the most relevant one wherein we can extract the expansion terms. This method to select the expansion terms needs more investigation and to be evaluated by several experiments.

Regarding the passage retrieval, the use of other kinds of Information Retrieval models like probabilistic model is another prospect of this work. We actually use the vectorial model based on the TF-IDF weighting function in our passage retrieval approach. We estimate that the use of other kinds of IR models should be investigated too. We aim at adapting our approach to use Okapi BM25 ranking function which is based on the probabilistic retrieval framework developed by [Robertson et al., 1994]. Use of other kinds of IR models eventually can improve the retrieval performance. Use of other kinds of IR model needs us to adapt our approach and to perform new tests.

We also consider studying the use of Language Modeling in passage retrieval in log files. Passage retrieval based on language model is studied in many work [Liu and Croft, 2002]. The basic idea is to retrieve the passages which have the most similar language model to the language model of the query. The first issue to study is the fact that although a query has an unique language model, log files generated by different tools may have different language models. Use of language model is however encouraged by the fact that log files are automatically generated documents which have pre-defined grammars.

6.2.2 Answer Extraction

In this thesis, the main focus was on the segmentation and afterwards on locating the answer in the log files, which leaded us to work on query improvement too. Although we are today able to locate the answers in the evolving and complex mass of log files, it needs to propose a solution in order to extract the answer in the retrieved passages.

For this purpose, we have been investigating an independent research project whose goal is to study different answer extraction approaches and propose a relevant one according to the specificities of log files.

According to the initial studies, two main directions are conceivable. First, the extraction of answers based on the patterns learned during a training phase should be investigated. Second, we need to study the relevance of semantic based methods which basically rely on the recognition of named entities.

Use of answer patterns is studied in question answering systems like as [Soubbotin, 2001]. This system uses a massive extraction patterns under the form of regular expressions and the exploitation of a factual knowledge database. Questions are analysed to determine the type and select the appropriate patterns to use. Each selected pattern is applied to the entire retrieved passages.

We are thus working on proposing an answer extraction module which relies on the use of answer patterns. We aims at proposing a system which learns or generates the patterns following a training phase. Indeed, although by means of our passage retrieval solution, the answer pattern can be less sophisticated, automatically generating them can still reduce the user interventions.

Because of the data specificities and mainly the different complex types of answer in the context of log files, the automatic generation of patterns needs more investigation. In this context, a whole table or an isolated element in a table can be considered as an answer. In addition, there are some issues in the structure of tables that need to be taken into account. For example, there are usually a heterogeneity between the structure of the header of a table and the structure of its data columns. As an instance, in log_A shown in Figure 1.4 on page 15, there is a table in lines 76 to 82. In this table, although we have five columns in the header, there are six data columns. Automatically generation of patterns for this table needs some considerations as the structure of the header do not correspond to that of data columns. In addition, there is a text line after the header which do not seems to be a table line. In such cases, we need to distinguish precisely the header from the data columns.

The other point to consider is the missing data in tables. We sometimes need to exactly locate an item in a table which can consist on missing data in different log files. The unexpected missing data in a table can make it difficult to locate an exact item in it by means of a pattern generated based on the table fulfilled entirely.

We also need to deal with the evolving aspect of log files. A user needs to be able to specify or generalize some elements in an answer too. For this purpose, we can initially ask the user to highlight the elements that should be generalized or specified. We can also develop a list of some elements that should be usually generalized or specified. For example, the numeric data should be usually generalized as they represent the variable values. However, the best solution is to generalize or specify elements based on the previous behaviours of the user. For this purpose, we need to implement an automatic learning phase.

Taking all this aspects into account, we are working on a system which automatically generates answer patterns following a training phase. This systems, at the current time, principally relies on the notion of regular expressions. The idea is to be able to generate patterns or improve the initially generated ones based on the findings of a training phase.

We are also interested to study the relevance of other types of answer extraction methods. Most of these methods are based on semantic analysis and recognition of named entities [Laurent et al., 2006]. This task refers to the extraction of atomic elements in texts which can be classified into predefined categories such as the names of persons, organizations, quantities, etc. Semantic analysis in the context of log files need more investigation. As described before, firstly, in this context, there is no semantic resource. Then, we are dealing with a kind of technical textual data wherein finding the semantic relation between terms is not trivial. Exploiting the semantic relations in the context of log files, recognition of the named entities of this domain, and building semantic resources are subsequently discussed in the next subsection.

6.2.3 Exploiting the New Knowledge of the Domain

As another perspective for this work, we aim at exploiting the semantic knowledge in the context of log files. We initially consider two main directions: (1) recognition of named entities in log files and (2) building the semantic resources of the domain.

First we would like to study the recognition of named entities in log files. As previously mentioned, it can be useful in information extraction in log files. Identifying the named entities can help to better generate the extraction patterns. According to the existing work in the domain of Named Entity Recognition (NER), the NER methods developed for one domain do not typically perform well on other domains [Poibeau and Kosseim, 2001]. In each domain, we can consider the special types of named entity.

BBN categories[1], proposed in 2002 by Ada Brunstein, consists of 29 types and 64 subtypes. These named entity categories are specially used for question answering task. We can note the main types in BBN categories as: Person name, person descriptor, NORP (nationality, religion, etc.), organization name, location name, product name, date, time, quantity, cardinal, event name, disease, etc. Each domain can possess its own named entity types. For example, in biological documents, we consider certain natural kind terms like biological species and substances as named entity types.

In the domain of log files, beside the general types of named entity, we need to define the new types. For example, the name of each design level, name of performed tests, and component names can be considered as the named entity types. We are also interested in named entity types like date, time, quantity, cardinal, and units of measurement.

Named entity recognition can be improved via a training phase. The ability to recognize previously unknown entities is an essential part of NER systems. The idea of supervised learning is to study the features of positive and negative examples of named entity over a large collection of annotated documents [Nadeau and Sekine, 2007]. We argue that NER in the domain of log files need more investigation notably providing a large collection of annotated log files to develop a NER system based on a supervised learning approach. In addition, we also need to study the recognition of the named entity types in this domain based on an expert analysis. As discussed in [Mohd et al., 2008], the identification of named entities differs between human and machine. The authors argue that the human analysis identifies What Named Entities more than a machine. The detection of named entities also relate to the content of the documents [Mohd et al., 2008]. Thus, a deep study is required to define this domain named entity types and to propose a NER system relevant to this domain and type of textual data.

1. http://www.ldc.upenn.edu/Catalog/docs/LDC2005T33/BBN-Types-Subtypes.html

Second, as another perspective of our future work in this direction, we consider the construction of the domain semantic resources. We previously have studied how to extract and validate the terminology of log files. We believe that a deeper analysis of semantic relations in the log files can be useful. Although the log files are not natural language texts, there should exist some semantic relations among the domain main terms.

Extraction of the domain terminology is the first and the main step towards a deeper exploitation of the semantic relation among the log file terms. Having the domain terminology, we can envisage to build the domain ontology. Acquisition of the domain ontology can be useful in different phases of information extraction in log files. Several approaches are based on the domain ontology to guide the information extraction process [Even and Enguehard, 2002]. [Silvescu et al., 2001] study ontology-assisted approaches to customizable data integration and Information Extraction from heterogeneous and distributed data sources. SOBA, presented by [Buitelaar et al., 2008], is an ontology-based Information Extraction system. It can be used to query information contained in different sources, including plain text and tables in an integrated and seamless manner.

Note that building the domain ontology needs a considerable amount of a domain expert time and knowledge beside adapting the NLP methods to this context.

Bibliography

E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *Proceedings of the 10th International Conference on World Wide Web*, WWW'01, pages 169–178, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. Cited pages 74 and 77.

A. Amrani, Y. Kodratoff, and O. Matte-Tailliez. A semi-automatic system for tagging specialized corpora. In *PAKDD*, pages 670–681, 2004. Cited page 113.

D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34:177–210, 1999. ISSN 0885-6125. Cited page 30.

D. Bernhard. Query expansion based on pseudo relevance feedback from definition clusters. In *Coling 2010: Posters*, pages 54–62, Beijing, China, August 2010. Coling 2010 Organizing Committee. Cited page 73.

D. Bourigault and C. Fabre. Approche linguistique pour l'analyse syntaxique de corpus. *Cahiers de Grammaire - Université Toulouse le Mirail*, (25):131–151, 2000. Cited pages 107 and 110.

E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 152–155, 1992. Cited page 113.

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-intensive question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC)*, pages 393–400, 2001. Cited page 69.

P. Buitelaar, P. Cimiano, A. Frank, M. Hartung, and S. Racioppa. Ontology-based information extraction and integration from heterogeneous data sources. *Int. J. Hum.-Comput. Stud.*, 66(11):759–788, 2008. ISSN 1071-5819. Cited page 152.

D. Buscaldi, P. Rosso, J. Gómez-Soriano, and E. Sanchis. Answering questions with an n-gram based passage retrieval engine. *Journal of Intelligent Information Systems*, 34: 113–134, 2010. Cited pages 34 and 64.

M. Caillet, J.-F. Pessiot, M. R. Amini, and P. Gallinari. Unsupervised learning with term clustering for thematic segmentation of texts. In *Proceedings of RIAO*, pages 648–656, 2004. Cited pages 32 and 35.

J. P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th annual International ACM SIGIR Conference on Research and development in information retrieval*, SIGIR'94, pages 302–310, New York, NY, USA, 1994. Springer-Verlag New York, Inc. Cited pages 27, 30, 35, 36, 37, and 58.

Y. Cao, F. Liu, P. Simpson, L. Antieau, A. Bennett, J. J. Cimino, J. Ely, and H. Yu. Askhermes: An online question answering system for complex clinical questions. *Journal of Biomedical Informatics*, 44(2):277 – 288, 2011. Cited page 37.

C. Carpineto, R. de Mori, G. Romano, and B. Bigi. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems*, 19:1–27, January 2001. Cited pages 74 and 77.

L. Carroll. Evaluating hierarchical discourse segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the As-*

sociation for Computational Linguistics, HLT'10, pages 993–1001, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. Cited page 30.

G. Chalendar, T. Dalmas, F. Elkateb-Gara, O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, L. Monceaux, I. Robba, and A. Vilnat. The question answering system qalc at limsi, experiments in using web and wordnet. In *TREC*, 2002. Cited pages 72 and 77.

F. Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics Conference*, pages 26–33, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. Cited page 30.

K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *Computational Linguistics*, volume 16, pages 22–29, 1990. Cited page 116.

C. L. A. Clarke, G. V. Cormack, and E. A. Tudhope. Relevance ranking for one to three term queries. *Information Processing and Management*, 36:291–311, January 2000. ISSN 0306-4573. Cited pages 71 and 77.

C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th annual International ACM SIGIR Conference on Research and development in information retrieval*, SIGIR'01, pages 358–365, New York, NY, USA, 2001. ACM. Cited page 69.

N. Collier, C. Nobata, and J. Tsujii. Automatic acquisition and classification of terminology using a tagged corpus in the molecular biology domain. *Journal of Terminology, John Benjamins*, 7(2):239–257, 2002. Cited pages 108 and 110.

H. Cui, R. Sun, K. Li, M.-Y. Kan, and T.-S. Chua. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual International ACM SIGIR Conference on Research and development in information retrieval*, SIGIR'05, pages 400–407, New York, NY, USA, 2005. ACM. Cited page 70.

B. Daille. *Approche mixte pour l'extraction automatique de terminologie : statistiques lexicales et filtres linguistiques*. PhD thesis, Universit Paris 7, 1994. Cited page 116.

B. Daille. Study and Implementation of Combined Techniques for Automatic Extraction of Terminology. In *The Balancing Act: Combining Symbolic and Statistical Approaches to Language, MIT Press*, pages 49–66, 1996a. Cited page 116.

B. Daille. Study and Implementation of Combined Techniques for Automatic Extraction of Terminology. In *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pages 49–66. The MIT Press, Cambridge, Massachusetts, 1996b. Cited page 87.

B. Daille. Conceptual structuring through term variations. In *Proceedings of the ACL 2003 workshop on Multiword expressions*, pages 9–16, Morristown, NJ, USA, 2003. Association for Computational Linguistics. Cited pages 109 and 115.

S. David and P. Plante. De la nécessité d'une approche morpho-syntaxique en analyse de textes. *Intelligence Artificielle et Sciences Cognitives au Québec*, 2(3):140–155, September 1990. Cited page 107.

H. Déjean, E. Gaussier, J. M. Renders, and F. Sadat. Automatic processing of multilingual medical terminology: applications to thesaurus enrichment and cross-language information retrieval. *Artificial Intelligence in Medicine*, 33:111–124, February 2005. Cited pages 105 and 108.

H. Doan-Nguyen and L. Kosseim. The problem of precision on restricted-domain question answering. In *Proceedings the ACL 2004 Workshop on Question Answering in Restricted Domains (ACL-2004)*, Barcelona, Spain, July 2004. ACL- 2004. Cited page 10.

T. Dorji, E.-s. Atlam, S. Yata, M. Fuketa, K. Morita, and J.-i. Aoe. Extraction, selection and ranking of field association (fa) terms from domain-specific corpora for building a comprehensive fa terms dictionary. *Knowledge and Information Systems*, 27:141–161, 2011. Cited pages 108 and 110.

M. Embarek. *Un système de question-réponse dans le domaine médical*. PhD thesis, Université Paris-Est, France, 2008. Cited page 37.

D. A. Evans and C. Zhai. Noun-phrase analysis in unrestricted text for information retrieval. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 17–24, Morristown, NJ, USA, 1996. Association for Computational Linguistics. Cited page 110.

F. Even and C. Enguehard. Extraction d'informations à partir de corpus dégradés. In *Proceedings of 9ème Conference sur le Traitement Automatique des Langues Naturelles (TALN'02)*, pages 105–115, 2002. Cited page 152.

F. M. Facca and P. L. Lanzi. Mining interesting knowledge from weblogs: a survey. *Data & Knowledge Engineering*, 53(3):225–241, 2005. ISSN 0169-023X. Cited page 11.

D. Ferrés and H. Rodríguez. Experiments adapting an open-domain question answering system to the geographical domain using scope-based resources. In *Proceedings of the Workshop on Multilingual Question Answering*, MLQA'06, pages 69–76, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. Cited page 69.

L. Gillard, P. Bellot, and M. El-Bèze. Influence de mesures de densité pour la recherche de passages et l'extraction de réponses dans un système de questions-réponses. In *CORIA'06*, pages 193–204. Université de Lyon, 2006. Cited pages 71 and 77.

R. Grishman. Natural language processing. *Journal of the American Society for Information Science*, 35(5):291–296, 1984. Cited page 9.

S. Guiasu. *Information Theory with Applications*. McGraw-Hill, New York, 1977. Cited page 87.

M. A. Hearst. Texttiling: segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23:33–64, March 1997. Cited page 31.

H. Hernault, H. Prendinger, D. A. duVerle, and M. Ishizuka. Hilda: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3), 2010. Cited page 36.

A. Ittycheriah and S. Roukos. Ibm's statistical question answering system-trec 11. In *TREC*, 2002. Cited pages 72 and 77.

B. J. Jansen. *Handbook of Research on Web Log Analysis*, chapter The Methodology of Search Log Analysis, pages 100–123. IGI Global, 2009. Cited page 11.

Z. M. Jiang, A. E. Hassan, G. Hamann, and P. Flora. An automated approach for abstracting execution logs to execution events. *Journal of Software Maintenance*, 20 (4):249–267, 2008. Cited pages 6 and 12.

A. K. Joshi. Natural language processing. *Science*, 253(5025):1242–1249, 1991. Cited page 9.

M. Kaisser. *Acquiring Syntactic and Semantic Transformations in Question Answering*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, 2009. Cited pages 72 and 77.

M.-Y. Kan, J. L. Klavans, and K. R. McKeown. Linear segmentation and segment significance. In *Proceedings of the 6th International workshop of very large corpora*, pages 197–205, 1998. Cited page 31.

M. Kaszkiel and J. Zobel. Passage retrieval revisited. In *Proceedings of the 20th annual International ACM SIGIR Conference on Research and development in information retrieval*, SIGIR'97, pages 178–185, New York, NY, USA, 1997. ACM. Cited pages 30, 35, and 63.

M. Kaszkiel and J. Zobel. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology*, 52:344–364, February 2001. ISSN 1532-2882. Cited pages 27, 30, 31, 35, 36, and 37.

M. Keikha, S. Gerani, and F. Crestani. Temper: a temporal relevance feedback method. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 436–447, Berlin, Heidelberg, 2011. Springer-Verlag. Cited page 64.

J.-U. Kietz, R. Volz, and A. Maedche. Extracting a domain-specific ontology from a corporate intranet. In *Proceedings of the 2nd workshop on Learning language in* *logic and the 4th Conference on Computational natural language learning - Volume 7*, ConLL'00, pages 167–175, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. Cited page 105.

G. Knorz. *Indexieren, Klassieren, Extrahieren.* Grundlagen der praktischen Information und Dokumentation. München, 1991. Cited page 106.

L. Kosseim and J. Yousefi. Improving the performance of question answering with semantically equivalent answer patterns. *Data Knowl. Eng.*, 66(1):53–67, 2008. ISSN 0169-023X. doi: http://dx.doi.org/10.1016/j.datak.2007.07.010. Cited pages 72 and 77.

H. Kozima. Text segmentation based on similarity between words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, ACL'93, pages 286–288, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics. Cited page 32.

A. Labadié and V. Prince. Lexical and semantic methods in inner text topic segmentation: A comparison between c99 and transeg. In *Proceedings of NLDB'08*, pages 347–349, Berlin, Heidelberg, 2008. Springer-Verlag. Cited page 32.

A. K. Lamjiri, J. Dubuc, L. Kosseim, and S. Bergler. Indexing low frequency information for answering complex questions. In *8th International Conference on Computer-Assisted Information Retrieval (RIAO'07)*, Carnegie Mellon University, Pittsburgh, PA, USA, 2007. Cited page 71.

D. Laurent, P. Séguéla, and S. Nègre. Cross lingual question answering using qristal for clef 2006. In *CLEF*, pages 339–350, 2006. Cited page 150.

E. H. Laurie, L. Gerber, U. Hermjakob, M. Junk, and C. yew Lin. Question answering in webclopedia. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 655–664, 2000. Cited pages 73 and 77.

G. G. Lee, J. Seo, S. Lee, H. Jung, B. hyun Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, H. Kim, and K. Kim. Siteq: Engineering high performance qa system using lexico-semantic pattern matching and shallow nlp. In *Proceedings of the Tenth Text REtrieval Conference (TREC)*, pages 442–451, 2001. Cited pages 71 and 77.

T. Li, F. Liang, S. Ma, and W. Peng. An integrated framework on mining logs files for computing system management. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge discovery in data mining*, KDD'05, pages 776–781. ACM, 2005. Cited page 6.

X. Li. A new robust relevance model in the language model framework. *Information Processing and Management*, 44:991–1007, May 2008. Cited pages 75 and 77.

X. Li and Z. Zhu. Enhancing relevance models with adaptive passage retrieval. In *Proceedings of the IR research, 30th European Conference on Advances in information retrieval*, ECIR'08, pages 463–471, Berlin, Heidelberg, 2008. Springer-Verlag. Cited pages 75 and 78.

M. Light, G. S. Mann, E. Riloff, and E. Breck. Analyses for elucidating current question answering technology. *Natural Language Engineering*, 7:325–342, December 2001a. Cited page 69.

M. Light, G. S. Mann, E. Riloff, and E. Breck. Analyses for elucidating current question answering technology. *Natural Language Engineering*, 7:325–342, December 2001b. ISSN 1351-3249. Cited pages 71 and 77.

D. Lin. Extracting collocations from text corpora. In *In First Workshop on Computational Terminology*, pages 57–63, 1998. Cited pages 108 and 110.

J. Lin. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems*, 25(2):6, 2007. Cited page 69.

X. Liu and W. B. Croft. Passage retrieval based on language models. In *Proceedings of the eleventh International Conference on Information and knowledge management*, CIKM'02, pages 375–382, New York, NY, USA, 2002. ACM. Cited page 148.

F. Llopis, A. Ferrndez, and J. Vicedo. Text segmentation for efficient information retrieval. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 13–29. Springer Berlin / Heidelberg, 2002a. Cited pages 27, 31, 35, 36, 37, and 58.

F. Llopis, J. L. Vicedo, and A. Ferrández. Passage selection to improve question answering. In *proceedings of the 2002 Conference on multilingual summarization and question answering - Volume 19*, MultiSumQA'02, pages 1–6, Stroudsburg, PA, USA, 2002b. Association for Computational Linguistics. Cited pages 28 and 35.

Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *Proceeding of the 18th ACM Conference on Information and knowledge management*, CIKM'09, pages 255–264, New York, NY, USA, 2009. ACM. Cited pages 65 and 75.

B. Magnini and R. Prevete. Exploiting lexical expansions and boolean compositions for web querying. In *Proceedings of the ACL-2000 workshop on Recent advances in natural language processing and information retrieval: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 11*, RANLPIR'00, pages 13–21, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. Cited pages 73 and 77.

C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. Cited page 73.

D. Marcu. *The theory and practice of discourse parsing and summarization*. MIT Press, 2000. Cited page 36.

C. meng Tan, Y. fang Wang, and C. do Lee. The use of bigrams to enhance text categorization. In *Information Processing and Management*, pages 529–546, 2002. Cited pages 109 and 115.

T. M. Mitchell. *Machine Learning*. McGraw-Hill International Edit, 1997. Cited page 51.

M. Mohd, F. Crestani, and I. Ruthven. A comparison of named entity patterns from a user analysis and a system analysis. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 679–683, Berlin, Heidelberg, 2008. Springer-Verlag. Cited page 151.

D. Moldovan, M. Pașca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21:133–154, April 2003. Cited page 72.

D. I. Moldovan and R. Mihalcea. Using wordnet and lexical operators to improve internet searches. *IEEE Internet Computing*, 4:34–43, January 2000. Cited page 105.

C. Monz. *From Document Retrieval to Question Answering*. PhD thesis, Universiteit van Amsterdam, Amsterdam, November 2003. Cited page 71.

D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30:3–26, January 2007. Cited page 151.

I. S. National and I. Soboroff. Overview of the trec 2004 novelty track, 2004. Cited page 90.

V. C. Nguyen, L. M. Nguyen, and A. Shimazu. Improving text segmentation with non-systematic semantic relation. In *Proceedings of the 12th International Conference on Computational linguistics and intelligent text processing - Volume Part I*, CICLing'11, pages 304–315, Berlin, Heidelberg, 2011. Springer-Verlag. Cited pages 32 and 33.

J. O'Connor. Retrieval of answer-sentences and answer-figures from papers by text searching. *Information Processing & Management*, 11(5-7):155 – 164, 1975. Cited page 63.

B. Ofoghi, J. Yearwood, and R. Ghosh. A semantic approach to boost passage retrieval effectiveness for question answering. In *ACSC'06: Proceedings of the 29th Australasian Computer Science Conference*, pages 95–101, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. ISBN 1-920682-30-9. Cited pages 27, 63, and 71.

M. Okabe and S. Yamada. Semisupervised query expansion with minimal feedback. *IEEE Transactions on Knowledge and Data Engineering*, 19:1585–1589, November 2007. Cited pages 75 and 77.

D. Paranjpe, G. Ramakrishnan, and S. Srinivasan. Passage scoring for question answering via bayesian inference on lexical relations. In *Proceedings of the 12th Text REtrieval Conference (TREC 2003)*, pages 305–310, 2004. Cited pages 73 and 77.

M. A. Pasca and S. M. Harabagiu. High performance question/answering. In *Proceedings of the 24th annual International ACM SIGIR Conference on Research and development in information retrieval*, SIGIR'01, pages 366–374, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6. Cited pages 73 and 77.

P. Pecina and P. Schlesinger. Combining association measures for collocation extraction. In *Proceedings of the COLING/ACL on Main Conference poster sessions*, COLING-ACL'06, pages 651–658, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. Cited page 109.

A. Penas, F. Verdejo, and J. Gonzalo. Corpus-based terminology extraction applied to information access. In *Proceedings of Corpus Linguistics 2001*, pages 458–465, 2001. Cited pages 107 and 110.

S. Petrović, J. Šnajder, and B. D. Bašić. Extending lexical association measures for collocation extraction. *Comput. Speech Lang.*, 24:383–394, April 2010. Cited pages 108, 109, and 110.

T. Poibeau and L. Kosseim. Proper name extraction from non-journalistic texts. *Language and Computers*, 37:144–157, 2001. Cited page 151.

J. M. Ponte and W. B. Croft. Text segmentation by topic. In *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 113–125, London, UK, 1997a. Springer-Verlag. ISBN 3-540-63554-8. Cited page 31.

J. M. Ponte and W. B. Croft. Text segmentation by topic. In *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 113–125, London, UK, 1997b. Springer-Verlag. Cited page 30.

J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0. Cited page 51.

R. E. Rice and C. L. Borgman. The use of computer-monitored data in information science and communication research. *Journal of the American Society for Information Science*, 34(4):247–256, 1983. Cited page 11.

F. Rinaldi, J. Dowdall, K. Kaljurand, M. Hess, and D. Mollá. Exploiting paraphrases in a question answering system. In *Proceedings of the second International workshop on Paraphrasing - Volume 16*, PARAPHRASE'03, pages 25–32, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. Cited page 18.

S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC*, 1994. Cited page 148.

J. Rocchio. *Book of Relevance Feedback in Information Retrieval*, pages 313–323. 1971. Cited page 74.

M. Roche, T. Heitz, O. Matte-Tailliez, and Y. Kodratoff. Exit: Un système itératif pour l'extraction de la terminologie du domaine à partir de corpus spécialisés. In *Proceedings of JADT'04 (International Conference on Statistical Analysis of Textual Data)*, volume 2, pages 946–956, 2004. Cited pages 107, 110, and 117.

M. Roche and Y. Kodratoff. Pruning terminology extracted from a specialized corpus for cv ontology acquisition. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4278 of *Lecture Notes in Computer Science*, pages 1107–1116. Springer Berlin / Heidelberg, 2006. Cited page 122.

M. Roche and Y. Kodratoff. Text and Web Mining Approaches in Order to Build Specialized Ontologies. *Journal of Digital Information*, 10(4):6, 2009. Cited page 87.

M. Roche and V. Prince. Managing the acronym/expansion identification process for text-mining applications. *International Journal of Software and Informatics, Special issue on Data Mining*, 2(2):163–179, 12 2008. ISSN 1673-7288. Cited pages 117 and 118.

G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987. Cited pages 82, 106, and 119.

G. Salton and C. Buckley. *Improving retrieval performance by relevance feedback*, pages 355–364. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. Cited page 65.

G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840. Cited page 82.

G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th annual International ACM SIGIR Conference on Research and development in Information Retrieval*, SIGIR'93, pages 49–58, New York, NY, USA, 1993. ACM. Cited page 36.

H. Saneifar, S. Bonniol, A. Laurent, P. Poncelet, and M. Roche. Terminology extraction from log files. In *Proceedings of 20th International Conference on Database and Expert Systems Applications*, DEXA'09, pages 769–776. Springer, 2009. Cited page 103.

H. Saneifar, S. Bonniol, A. Laurent, P. Poncelet, and M. Roche. Recherche de passages pertinents dans les fichiers logs par enrichissement de requêtes. In *Proceedings of 7th French Information Retrieval Conference*, CORIA'10, pages 239–254. Centre de Publication Universitaire, 2010a. Cited page 61.

H. Saneifar, S. Bonniol, A. Laurent, P. Poncelet, and M. Roche. Passage retrieval in log files: an approach based on query enrichment. In *Proceedings of Advances in Natural Language Processing, 7th International Conference on NLP*, IceTAL'10, pages 357–368, Berlin, Heidelberg, 2010b. Springer-Verlag. Cited page 61.

H. Saneifar, S. Bonniol, A. Laurent, P. Poncelet, and M. Roche. How to rank terminology extracted by exterlog. In A. Fred, J. L. G. Dietz, K. Liu, and J. Filipe, editors, *Knowledge Discovery, Knowlege Engineering and Knowledge Management*, volume 128 of *Communications in Computer and Information Science*, pages 121–132. Springer Berlin Heidelberg, 2011a. Cited pages 103 and 116.

H. Saneifar, S. Bonniol, P. Poncelet, and M. Roche. Identification des divisions logiques de fichiers logs. In *Proceedings of 18èmes Rencontres de la Société Francophone de Classification*, SFC'11, Orleans, France, 2011b. Cited page 25.

D. Schiffrin. *Discourse markers*. Cambridge University Press, Cambridge, 1987. Cited page 36.

E. W. Selberg. Information retrieval advances using relevance feedback, 1997. URL http://selberg.org/~speed/papers/generals/generals.pdf. Cited page 65.

A. Silvescu, J. Reinoso-castillo, and V. Honavar. Ontology-driven information extraction and knowledge acquisition from heterogeneous, distributed, autonomous biological data sources. In *Proceedings of the IJCAI-2001 Workshop on Knowledge Discovery from Heterogeneous, Distributed, Autonomous, Dynamic Data and Knowledge Sources*, 2001. Cited page 152.

F. Smadja, K. R. McKeown, and V. Hatzivassiloglou. Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 22(1):1–38, 1996. Cited page 116.

F. Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1): 143–177, 1993. ISSN 0891-2017. Cited pages 109 and 110.

M. M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Text REtrieval Conference*, 2001. Cited page 149.

M. Sun and J. Y. Chai. Discourse processing for context question answering based on linguistic knowledge. *Knowledge-Based Systems*, 20:511–526, August 2007. Cited pages 30 and 36.

C.-M. Tan, Y.-F. Wang, and C.-D. Lee. The use of bigrams to enhance text categorization. *Information Processing and Management*, 38:529–546, July 2002. ISSN 0306-4573. Cited page 44.

O. Tarek. La segmentation des documents techniques en amont de l'indexation : définition d'un modèle. *Revue d'Information Scientifique et Technique (RIST)*, vol. 13 (no1):79–94, 2003. Cited page 30.

S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual*

International ACM SIGIR Conference on Research and development in informaion retrieval, SIGIR'03, pages 41–47, New York, NY, USA, 2003a. ACM. ISBN 1-58113-646-3. Cited pages 71 and 72.

S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual International ACM SIGIR Conference on Research and development in informaion retrieval*, SIGIR'03, pages 41–47, New York, NY, USA, 2003b. ACM. Cited pages 63, 83, and 87.

B. I. Thattil, Jiny Antony [IN]. Context-based information retrieval. Patent Application, 08 2008. US 2008/0201350 A1. Cited page 12.

J. Tiedemann and J. Mur. Simple is best: experiments with different document segmentation strategies for passage retrieval. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, IRQA'08, pages 17–25, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. Cited pages 28, 35, 38, and 58.

J. Tiedemann. Comparing document segmentation strategies for passage retrieval in question answering. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP'07)*, Borovets, Bulgaria, 2007. Cited page 69.

P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL'01, pages 491–502, London, UK, 2001. Springer-Verlag. ISBN 3-540-42536-5. URL http://dl.acm.org/citation.cfm?id=645328.650004. Cited page 117.

J. Valdman. Log file analysis. Technical report, Department of Computer Science and Engineering (FAV UWB), 2001-04. Cited pages 6 and 11.

L. van der Plas and J. Tiedemann. Using lexico-semantic information for query expansion in passage retrieval for question answering. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, IRQA'08, pages 50–57, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. Cited pages 64 and 81.

C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979. ISBN 0-408-70929-4. Cited page 87.

V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998. Cited page 76.

E. M. Voorhees. The trec-8 question answering track report. In *Proceedings of TREC-8*, pages 77–82, 1999. Cited page 93.

C. Wade and J. Allan. Passage retrieval and evaluation. Technical report, 2005. Cited page 63.

M. A. Walker. Redundancy in collaborative dialogue. In *Proceedings of the 14th Conference on Computational linguistics - Volume 1*, COLING'92, pages 345–351, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. Cited page 31.

H. F. Witschel. Terminology extraction and automatic indexing - comparison and qualitative evaluation of methods. In *Proc. of Terminology and Knowledge Engineering (TKE)*, 2005. Cited pages 105 and 106.

J. Wu, I. Ilyas, and G. Weddell. A study of ontology-based query expansion. Technical report, University of Waterloo, 2011. Cited page 74.

J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual International ACM SIGIR Conference on Research and development in information retrieval*, SIGIR'96, pages 4–11, New York, NY, USA, 1996. ACM. Cited page 76.

J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, 18:79–112, January 2000. Cited pages 74, 77, and 78.

K. Yamanishi and Y. Maruyama. Dynamic syslog mining for network failure monitoring. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge discovery in data mining (KDD'05)*, pages 499–508, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. Cited page 11.

H. Yang and T.-S. Chua. The integration of lexical knowledge and external resources for question answering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC'2002)*, pages 155–161, Maryland, USA, 2002. Cited page 73.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th Conference on Computational linguistics - Volume 2*, COLING'00, pages 940–946, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. Cited page 105.

J. Zobel, A. Moffat, R. Wilkinson, and R. Sacks-Davis. Efficient retrieval of partial documents. In *Proceedings of the second Conference on Text retrieval Conference*, pages 361–377, Elmsford, NY, USA, 1995. Pergamon Press, Inc. Cited page 35.

List of Figures

List of Tables

Abstract

Nowadays, modern computing systems are instrumented to generate huge reports in the format of log files. Log files are generated to report the status of systems, products, or even causes of problems that can occur. Analysing log files, as an attractive approach for automatic system management and monitoring, has been enjoying a growing amount of attention [Li et al., 2005]. Log file analysis could be a tremendous task that requires enormous computational resources, long time and sophisticated procedures. In this thesis, we present contributions to the challenging issues which are encountered in question answering and locating information in a special kind of log files, i.e., those generated by Electronic Design Automation (EDA) tools. EDA is a category of software tools for designing electronic systems such as Integrated Circuits (IC). Knowing that these designs tools usually generate gigabytes of log files each day, the problem is to wade through all of this data to locate the critical information we need to verify the quality check rules. Manually locating information is a tedious and cumbersome process. Furthermore, Automated analysis of such logs is complex due to their heterogeneous and evolving structures and the large non-fixed vocabulary. In this thesis, by each contribution, we answer to questions raised in this work due to the data specificities or domain requirements. We investigate the main concern "how the specificities of log files can influence the information locating and natural language processing methods?". We present different contributions as fellow: > Proposing a novel method to recognize and identify the logical units in the log files to perform a segmentation according to their structure. We thus introduce a method to characterize complex logical units found in log files by proposing an original type of descriptor to model the textual structure of text documents. > Proposing an approach to locate the requested information in the log files based on passage retrieval. To improve the performance of passage retrieval, we propose a novel query expansion approach, based on two relevance feedback steps, to overcome the difficulties like mismatch vocabularies. In the first step, we determine the explicit relevance feedback by identifying the context of questions. The second step, consisting of a novel type of pseudo relevance feedback, uses a new term weighting function, called TRQ (Term Relatedness to Query), introduced in this work, which gives a score to the terms of corpus according to their relatedness to the query. We also investigate how to apply our approach to documents from general domains. > Studying the use of morpho-syntactic knowledge in our approaches. We here introduce our approach, named Exterlog (EXtraction of TERminology from LOGs), to extract the terminology of log files and evaluate their relevancy using the statistical measures adapted to this context.

Keywords: *Information Retrieval, Natural Language Processing, Text Mining, Question Answering Systems, Complex Textual Data, log files*

Résumé

De nos jours, les systèmes informatiques sont instrumentés pour produire des rapports, dans un format textuelles, généralement appelé fichiers log. Les fichiers logs représentent la source principale d'informations sur l'état des systèmes, des produits, ou encore les causes de problèmes qui peuvent survenir. Analyse de ces fichiers pourrait être une tâche difficile qui exige d'énormes ressources de calcul, de temps et de procédures sophistiquées. Dans cette thèse, nous nous concentrerons sur un type des fichiers logs générés par des systèmes EDA (Electronic Design Automation) qui contiennent des informations sur la configuration et la conception des Circuits Intégrés (CI) ainsi que les tests de vérification effectués sur eux. Ces informations, très peu exploitées actuellement, sont particulièrement attractives et intéressantes pour la gestion de conception, la surveillance et surtout la vérification de la qualité de conception. Cependant, la complexité de ces données textuelles complexes rend difficile l'exploitation de ces connaissances. Au sein de cette thèse, nous étudions principalement "comment les spécificités de fichiers logs peuvent influencer l'extraction de l'information et les méthodes de TALN?". Le problème est accentué lorsque nous devons également prendre leurs structures évolutives et leur vocabulaire spécifique en compte. Ainsi, les contributions de cette thèse consistent brièvement en : > Proposer une méthode d'identification et de reconnaissance automatique des unités logiques dans les fichiers logs afin d'effectuer une segmentation textuelle selon la structure des fichiers. Au sein de cette approche, nous proposons un type original de descripteur qui permet de modéliser la structure textuelle et le layout des documents textuels. > Proposer une approche de la localisation de réponse (recherche de passages) dans les fichiers logs. Afin d'améliorer la performance de recherche de passage ainsi que surmonter certains problématiques tel que "mismatch vocabulary", nous proposons une approches d'enrichissement de requêtes. Cette approches, fondée sur la notion de relevance feedback, consiste en un processus d'apprentissage et une mesure original de pondération des mots pertinents du contexte qui sont susceptibles d'exister dans les passage adaptés. Cette mesure, appelée TRQ (Term Relatedness to Query), a pour objectif de donner un poids élevé aux termes qui ont une probabilité importante de faire partie des passages pertinents. Cette approche est également adaptée et évaluée dans les domaines généraux. > Etudier l'utilisation des connaissances morpho-syntaxiques au sein de nos approches. A cette fin, nous proposons la méthode Exterlog, adaptée aux spécificités des logs, pour en extraire la terminology.

Mots clefs : *Recherche d'information, Traitement de la langue naturelle, Fouille de textes, Système question réponse, Données textuelles complexes, Fichiers logs*

LIRMM — 161, rue Ada — 34095 Montpellier cedex 5 — France