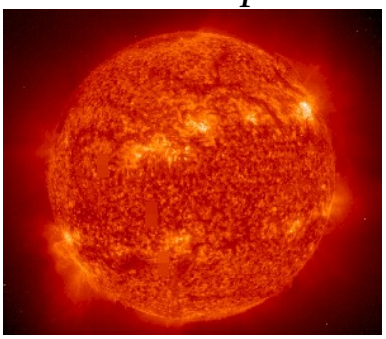
Processing Robustly Natural Language

The Chaos Experience



User's Survival Guide

Roberto Basili **Daniele Pighin**
Fabio Massimo Zanzotto
University of Rome Tor Vergata,
Department of Computer Science, Systems and Production,
00133 Roma (Italy),
chaos@info.uniroma2.it

Contents

Acknowledgments

This work would not have been possible without many students and researchers that contributed to the development of the different versions of Chaos since 1995 at the AI and NLP laboratory of the DISP (Department of Computer Science, Systems and Production) of the University of Roma, Tor Vergata. Our thanks go to all of them. However, the main acknowledge goes to prof. Maria Teresa Pazienza that encouraged and supported the researches on CHAOS for all these years.

1 Chaos Principles in a nutshell

Chaos is a modular and lexicalized syntactic and semantic parser for Italian and for English. It uses the eXtended Dependency Graph (XDG) as syntactic formalism as it well represents alternative syntatic interpretation. The system is thus defined as a cascade of processing modules ($P_1, ..., P_n$), via composition of processors:

$$Chaos : \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma'\Delta'}$$

with

$$Chaos(xdg) = P_n \circ P_{n-1} \circ \cdots \circ P_2 \circ P_1(xdg)$$

The system offers a collection of modules for designing parsing architectures. The pool of modules consists of:

- a tokenizer (TOK), matching words from character streams
- a morphologic analyser (MOA) that attaches (possibly ambiguous) syntactic categories and morphological interpretations to each word and matches named entities existing in catalogues
- a named entities matcher (NER, NES) that recognizes complex named entities according to special purpose grammars
- a rule-based part-of-speech tagger (POS)
- a POS disambiguation module (PMF) that resolves potential conflicts among the results of the POS tagger and the morphologic analyser
- a chunker (CHK)
- a verb argument detector (VSA)
- a shallow syntactic analyser (SSA)

2 How to ... run the *chaos* parser

This chapter describes how to install and run the Chaos Parser for the Italian and for the English language. Chaos can be used in three ways: in a monolithic version, in a client-server version, and with the graphical user interface.

2.1 How to install the *chaos* parser

Unzip the choas package, run `install` and follow the instructions.

2.2 How to ... run the Monolithic Version

Chaos is a single process that follows this cicle: loading of the knwoledge bases, analysis of input text or texts, and release of the knowledge bases. This command is useful when large collections of documents have to be processed. The commant to access this configuration is: `chaosparser`.

This message is displayed typing `chaosparser -[?|help|h]`:

```
chaosparser <input> [output] [-l <it|en>] [-k
          <kbname>] [-m <modules>]
```

where `input` (mandatory) can be any of:

`-if <input file>`

`-it <input text (quoted)>`

`-id <input directory>`

and output (optional) can be etiher

`-of <output file>` (used with `-if` or `-it`) [default `chaos.out`]

`-od <output dir>` (used with `-id`) [default `chaos.out.dir`]

you can also select the output format (xml, prolog, or serialized object) using

`-ot <xml|pl|obj>` [default `xml`]

other options:

`-l` select input language (italian (it) or english (en)) [default `en`]

`-k` select which knowledge base to use [default `default`]

`-p` provide the list of processors to be run. The list is a comma separated string with no blanks. Available processors are: `INN, TOK, MOA, POS, PMF, TEM, NER, NES, CHK, VSA, SSA`

2.3 How to ... run the Client-Server Version

Chaos is two processes, one server (the parser) and one client that loads the files to parse. The cicle to be followed is this: initialize the server (`chaosserver` on a given port and then run the client (`chaosclient`) as many times as needed.

2.3.1 Chaos Server

The server is launched with the command `chaosserver`.

This message is displayed typing `chaosserver -[?|help|h]`:

```
chaosserver [-l <it|en>] [-p <portno>] [-b <backlog>]
```

default values are:

language [en]

port number [3333]

backlog [5]

2.4 Chaos Client

The server is launched with the command `chaosclient`.

This message is displayed typing `chaosclient -[?|help|h]`:

```
chaosclient [options] -t|--text <text to parse>
```

where options are:

`[-i|--insensitive]` ignore input text case (by default it doesn't)

`[-h|--host <hostname>]` select server host (defaults to '`localhost`')

`[-p|--port <portno>]` select server port number (defaults to '`3333`')

`[-o|--out <filaneme>]` file to write output to (defaults to '`./out.cha`')

`[-f|--format <xml|pl|obj>]` output format (defaults to '`xml`')

`[-m|--modules <list of modules>]` select modules (processors) to be run (defaults to '`INN, TOK, MOA, POS, PMF, TEM, NER, NES, CHK, VSA, SSA`')

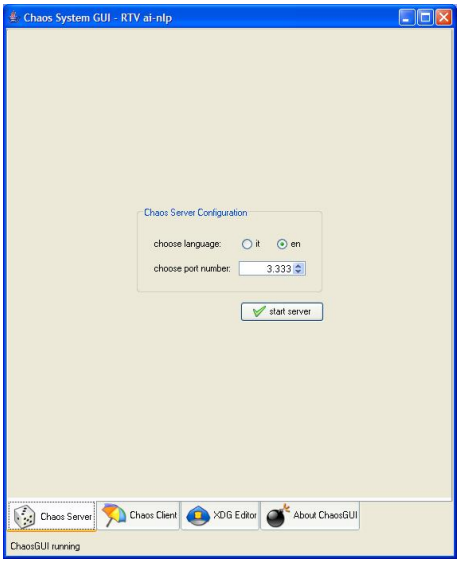
2.5 How to ... run and use the Graphical User Interface

The Chaos graphical user interface is based on the client-erver architecture where the GUI mainly plays the roles of client and of server launcher. The main feature of the GUI is the possibility of observing the eXtended Dependendy Graphs, i.e. the grammatical representations of input sentences in texts. These structures are shortly described in Sec. 3.

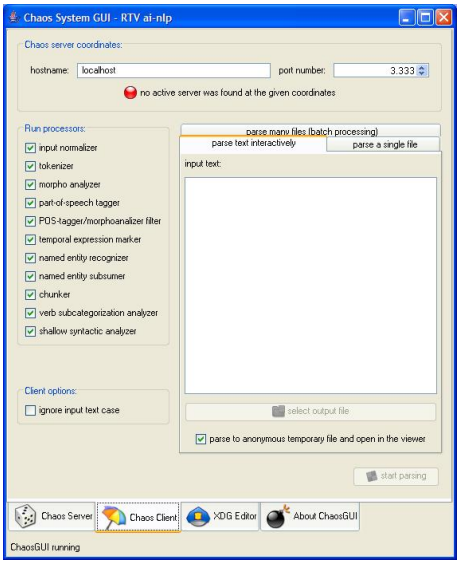
Run `chaosgui` to access the Chaos GUI.

The aspect of the Chaos GUI is costantly changing. However, it will always give the possibility of:

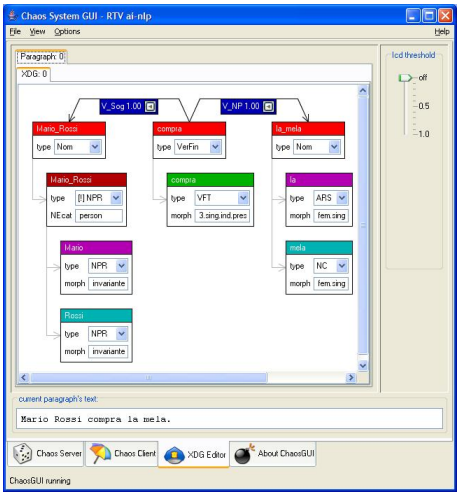
- launching a chaos server



- processing a text, a file, or a file directory



- observing the XDGs of a given file



3 How to .. interpret syntactic representations

This chapter describes the formalism to represent the syntactic information for each sentence and the grammatical information that is expected to be extracted for English and for Italian.

3.1 eXtended Dependeny Graph (XDG)

Syntactic interpretations are represented using the eXtended dependency graphs (XDG). This formalism is a mixture of dependencies and constituents. In particular, it is a dependency graph whose nodes C are *constituents* and whose edges ICD are the *grammatical dependencies* among the constituents, i.e.

$$\mathcal{XDG} = (C, ICD)$$

Each node is a complete tree whose nodes are feature structures. The XDG formalism efficiently models the syntactic ambiguity. In general, alternative interpretations for dependencies are represented by alternative $d \in ICD$. A useful property can be imposed on xdgs to select a single (partial) syntactic interpretation. A *planar xdg* is a single (although possibly partial) syntactic reading.

An example of XDG seen with the graphical user interface is given in Fig. 1
An XDG can be accessed in four different ways:

- as a Java object
- as a XML document

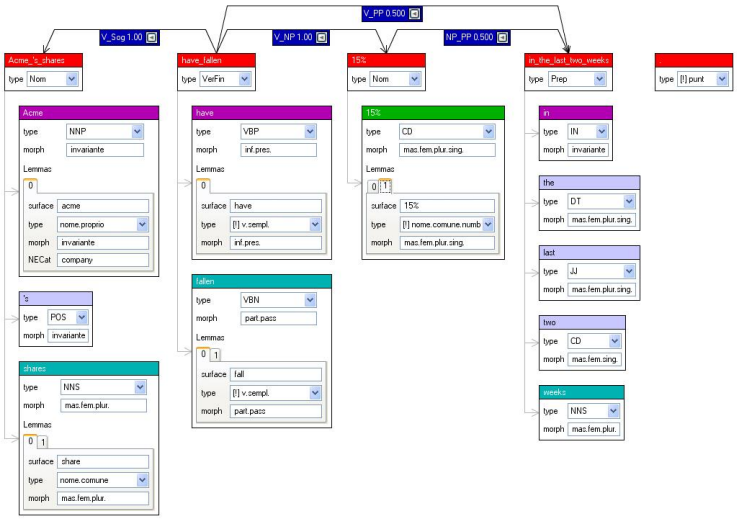


Figure 1: An eXtended Dependency Graph

- as a prolog atom
- using the graphical user interface

3.2 English Grammatical Theory

This section describes the syntactic categories used for the simple constituents, the complex constituents, and the inter-constitutent depencencies for the English syntactic representation.

3.2.1 Constituent categories

Simple constituent categories follow the definition of the Penn Treebank categories whilst complex constituent categories follow in the table.

Categories for Complex Constituents

Class	Description
Agg	Adjectival Chunk
Avv	Adverbial Chunk
CongCo	Coordinative Conjunction Chunk
CongSub	Subordinative Conjunction Chunk
Nom	Nominal Chunk
Prep	Prepositional Chunk
VerFin	Finite Verbal Chunk
VerGer	-ing Verbal Chunk
VerInf	Infinite Verbal Chunk
VerPart	Past Paricle Verbal Chunk
VerPred	Predicative Adjectival Chunk
VerNom	Nominal Verbal Chunk
VerPrep	Preposiotional Verbal Chunk
?	Unknown Chunk

3.3 Inter-Constitent Dependency Categories over Chunk Types

Label	Grammatical Dependency Category
V_Sog	Grammatical Subject
V_Obj	Grammatical Object
V_NP	Indirect Object
V_PP	Verb Preposition Modifier
V_Adv	Verb Adverb Modifier
NP_PP	Noun Prepositional Modifier
PP_PP	Noun Prepositional Modifier (originated from a prepositional chunk)
NP_Adj	Noun Adjectival Modifier
PP_Adj	Noun Adjectival Modifier (originated from a prepositional chunk)
NP_VPart	Noun Verb Past Particle Modifier
PP_VPart	Noun Verb Past Particle Modifier (originated from a prepositional chunk)
Adj_PP	Adjective Prepositional Modifier
Adv_PP	Adverb Prepositional Modifier

3.4 Italian Grammatical Categories

This section describes the syntactic categories used for the simple constituents, the complex constituents, and the inter-constitutent depencencies for the Italian syntactic representation.

3.4.1 Constituent categories

Categories for Simple Constituents (POS tags)

Internal Code	**Explanation**
AGS	Aggettivo Singolare
AGP	Aggettivo Plurale
ADS	Aggettivo Determinativo Singolare
ADP	Aggettivo Determinativo Plurale
AGI	Aggettivo Interrogativo
AGV	?
NUM	Numero
ARS	Articolo Singolare
ARP	Articolo Plurale
AVV	Avverbio
CO	Congiunzione
COA	Congiunzione Avverbiale
CPU	Congiunzione Punto
COP	Congiunzione Parentesi
COS	Congiunzione Subordinativa
DAT	Data
PR	Pronome
PRN	Pronome Interrogativo
PSG	Pronome Singolare
PPL	Pronome Plurale
PRR	Pronome Relativo
NC	Nome Comune
NCS	Nome Comune Singolare
NCP	Nome Comune Plurale
NPR	Nome Proprio
PSE	Preposizione Semplice
PAS	Preposizione Articolata Singolare
PAP	Preposizione Articolata Plurale
PIM	Preposizione Impropria
VX	Verbo Ausiliare
VFT	Verbo Finito Transitivo
VFI	Verbo Finito Intransitivo
VNT	Verbo Non Finito Transitivo
VNI	Verbo Non Finito Intransitivo
VNP	Verbo Non Finito Transitivo Participio Passato
VIP	Verbo Non Finito Intransitivo Participio Passato
VTR	Verbo Transitivo
VIN	Verbo Intransitivo
SYM	Simbolo

Categories for Complex Constituents

Class	Description
Agg	Adjectival Chunk
Avv	Adverbial Chunk
CongCo	Coordinative Conjunction Chunk
CongSub	Subordinative Conjunction Chunk
Nom	Nominal Chunk
Prep	Prepositional Chunk
VerFin	Finite Verbal Chunk
VerGer	-ing Verbal Chunk
VerInf	Infinite Verbal Chunk
VerPart	Past Paricle Verbal Chunk
VerPred	Predicative Adjectival Chunk
VerNom	Nominal Verbal Chunk
VerPrep	Preposiotional Verbal Chunk
?	Unknown Chunk

3.5 Inter-Constitent Dependency Categories over Chunk Types

Label	Grammatical Dependency Category
V_Sog	Grammatical Subject
V_Obj	Grammatical Object
V_NP	Indirect Object
V_PP	Verb Preposition Modifier
V_Adv	Verb Adverb Modifier
NP_PP	Noun Prepositional Modifier
PP_PP	Noun Prepositional Modifier (originated from a prepositional chunk)
NP_Adj	Noun Adjectival Modifier
PP_Adj	Noun Adjectival Modifier (originated from a prepositional chunk)
NP_VPart	Noun Verb Past Particle Modifier
PP_VPart	Noun Verb Past Particle Modifier (originated from a prepositional chunk)
Adj_PP	Adjective Prepositional Modifier
Adv_PP	Adverb Prepositional Modifier

4 How to ... navigate the Java Documentation

If you want to integrate the chaos processor in one of your applications or you want to integrate a new module you cannot avoid to go into the Java Api Documentation

that you find enclosed with the system. There are two main packages to know:

- the chaos.XDG package that describes how an eXtended Dependency Graph is implemented
- the chaos.processors packace that describes how processors are organized and implemented

Good luck! For any inconvenient please contact chaos@info.uniroma2.it