CoLing 2000

Tutorial

Trends in Robust Parsing

Jacques Vergne
GREYC - Université de Caen
France

Introduction

• Aim of the course :

- giving you an overview of the field,

while stressing the evolution of concepts

• Aim of the practical :

- giving you a practical entrance into the field
- bringing a concrete basis to the course

2 meanings of ***parsing***

• ***parsing*** with formal grammars (HPSG, LFG, TAG, ...)
to be compared with robust parsing

• tagging, chunking, partial, shallow, or robust ***parsing***
here is the topic of this course

these two meanings correspond to 2 different paradigms
inside the NLP community

Outline of the course

1. Standard operations in robust parsing

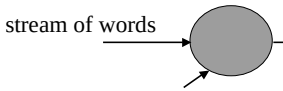
operation	*output unit*
• 1.1. part-of-speech tagging	word
• 1.2. chunking	chunk
• 1.3. clause bracketing	clause

1.1. Part-of-speech tagging

Overview :

… the agency issued
the inspection order ...

… the<det> agency<N> issued<V>
the<det> inspection<N> order<N> ...



how to tag words
= dictionary
+ contextual deduction rules

1.1. Part-of-speech tagging

What for ?

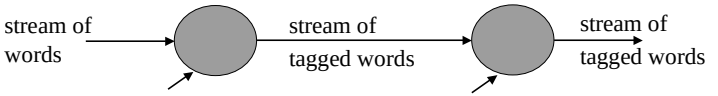
- for shallow parsing on raw material
- or to replace morpho-lexical analysis
  before classic syntactic analysis
  to make it less combinatorial

1.1. Part-of-speech tagging

Standard method :

looking in the dictionary choosing a tag thanks to the **context**

… the order ... … the<det> order<N,V> ... … the<det> order<N> ...



dictionary
order <N,V>
the <det>

contextual deduction rule
<det> + <N,V> => <N>

1.1. Part-of-speech tagging : contextual deduction rules

3 ways to build contextual deduction rules :

•1• extracting tag contiguity frequencies from hand-tagged corpora
Debili 1977, Church 1988 and 1993, Merialdo 1994, ...

•2• extracting symbolic rules from hand-tagged corpora
Brill tagger

•3• manually writing symbolic rules
Xerox Grenoble (Chanod et al.), GREYC Caen (Vergne et al.)

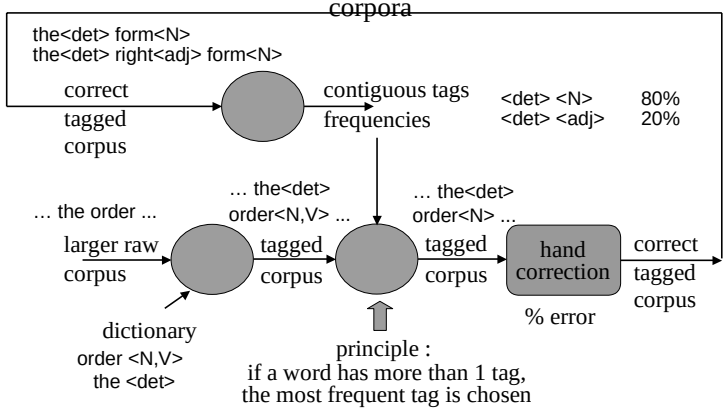
1.1. Part-of-speech tagging : contextual deduction rules

3 ways to build contextual deduction rules :

	extracting from hand-tagged corpora ...	manually writing ...
… tag contiguity frequencies	•1• Debili 1977, Church 1988 and 1993, Merialdo 1994, ...	
… symbolic rules	•2• Brill tagger	•3• Xerox Grenoble, GREYC Caen

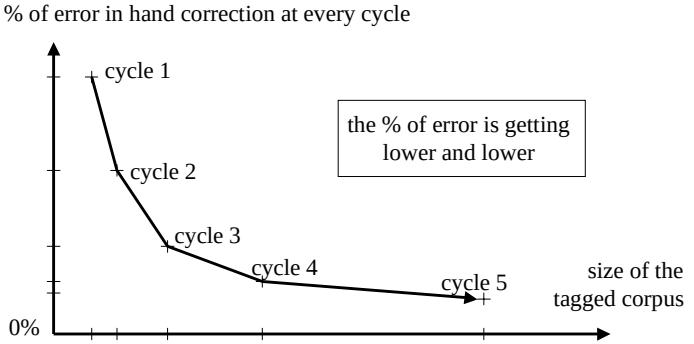
1.1. Part-of-speech tagging :

•1• extracting tag contiguity frequencies from hand-tagged corpora



1.1. Part-of-speech tagging :

•1• extracting tag contiguity frequencies from hand-tagged corpora

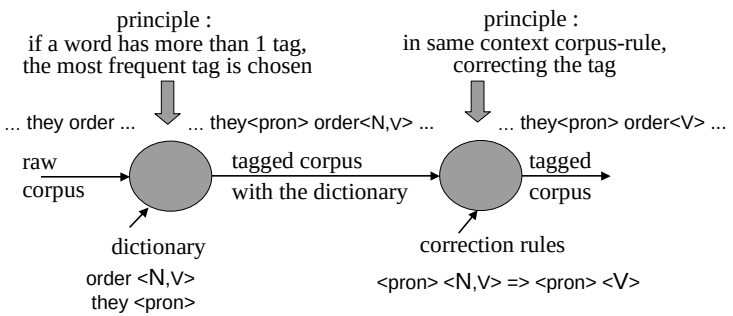


1.1. Part-of-speech tagging :

•2• extracting symbolic rules from hand-tagged corpora

Brill tagger

The tagging process :

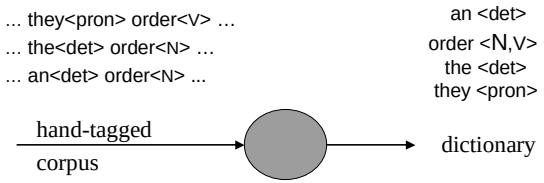


1.1. Part-of-speech tagging :

•2• extracting symbolic rules from hand-tagged corpora

Brill tagger

Building the dictionary :

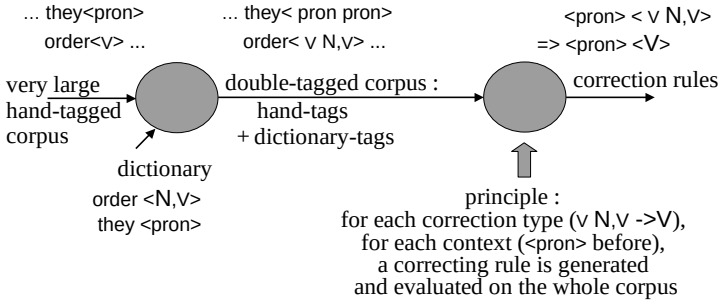


1.1. Part-of-speech tagging :

•2• extracting symbolic rules from hand-tagged corpora

Brill tagger

Extracting correction rules (= training the tagger) :



1.1. Part-of-speech tagging :

•3• manually writing symbolic rules : rule-based systems

- Xerox Research Centre Europe- Grenoble (XRCE)

  "Regular Expressions for Language Engineering"

  replace all <regular expression 1> by <regular expression 2>

  to insert markers in the input string, to filter the input string

- GREYC - Caen : engine + rules

  conditions on the current token and its context
  => actions on the current token and its context

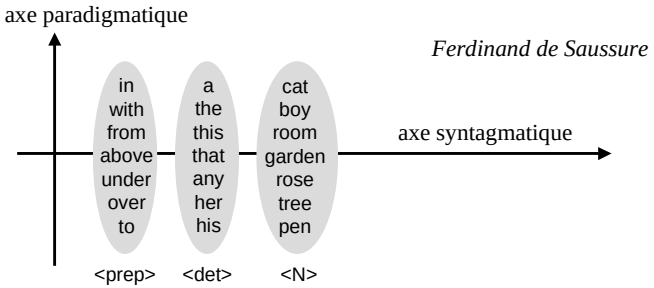
1.1. Part-of-speech tagging :

importance of the tagset

- what are parts of speech ?
  - a traditional tagset coming from Greek rhetoric passed on by Latin grammar
  - a way to categorize words at school
- is this tagset adequate for automatic tagging ?
- the tagset should catch regularities of tag contiguities
- we can find and test other tagsets

1.1. Part-of-speech tagging :

importance of the tagset



principle : **a tag** <--> a paradigm, to catch regularities of tag contiguities

1.1. Part-of-speech tagging :

3 ways to connect dictionary to symbolic contextual rules

- reductionist deduction : (constraints grammars, Helsinki)
  in this context, this token can't get that tag : <det> <N,V> => discard V
  => every token must be in the dictionary, with all its possible tags
  this double exhaustiveness isn't realistic

- constructive deduction :
  in this context, every token gets that tag : <det> <?> => add N
  => a token may not be in the dictionary or some tags may be missing
  this is a robust method : a natural language can't be totally defined

- the most frequent tag by default is put in the dictionary : will<V>
  the other tags deduced by constructive deduction : her<det> will<V> => will<N>

1.1. Part-of-speech tagging :

the tagging process : triggering rules on tokens

- algorithm of the tagging process :

```
for each current token
   for each rule
      if the rule may be applied to the current token and its context
         then apply the rule
```

- linear complexity, constant and foreseeable rate : n tokens / s
- the beginning of a renewal in parsing strategies

Outline of the course

- 1. Standard operations in robust parsing
  - 1.1. Tagging
  - 1.2. **Chunking**
  - 1.3. Clause bracketing
- 2. Shared properties, and differences in robust parsing
- 3. Two technologies to implement symbolic rules
  - 3.1. Finite-State Transducers (FST)
  - 3.2. Engine and rules
- 4. Typical applications
- 5. Comparing robust parsing with formal grammar parsing
- 6. Introduction to the practical

1.2. Chunking

the concept of chunk from Abney 91 in "Parsing by Chunks"

an example :

[I begin] [with an intuition] : [when I read] [a sentence] ,
[I read it] [a chunk] [at a time] .

a prosodic segment :

These chunks correspond in some way to prosodic patterns.

[...] the strongest stresses in the sentence fall one to a chunk, and pauses are most likely to fall between chunks.

1.2. Chunking

the concept of chunk from Abney 91 in "Parsing by Chunks"

an example :

[I begin] [with an intuition] : [when I read] [a sentence] ,
[I read it] [a chunk] [at a time].

internal structure :

The typical chunk consists of a single content word surrounded by a constellation of function words, matching a fixed template.

1.2. Chunking

the concept of chunk from Abney 91 in "Parsing by Chunks"

words within a chunk :

The typical chunk consists of a single content word surrounded by a constellation of function words, matching a fixed template.

A simple context-free grammar is quite adequate to describe the structure of chunks.

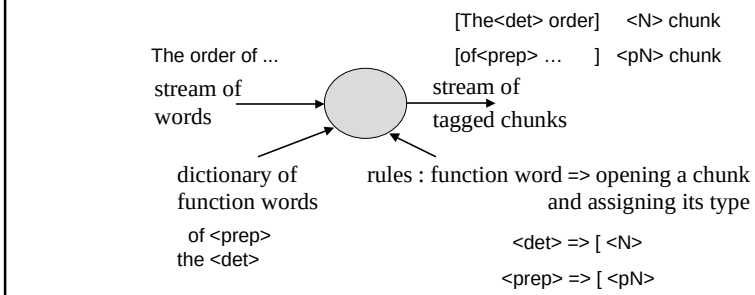
chunks within a sentence :

By contrast, the relationships between chunks are mediated more by lexical selection than by rigid templates.
[...] the order in which chunks occur is much more flexible than the order of words within chunks.

1.2. Chunking

functions of chunking : delimiting and tagging chunks

how does it work ? (the GREYC method)



1.2. Chunking

- lexical resources :
  beginnings (function words), endings of chunks
  separators (punctuation) of chunks

- the chunking process :
  as in tagging, triggering rules on tokens

1.2. Chunks & tagging words

some more features of the contextual deduction at word level

2 kinds of contiguities : [I • begin] • [with • an • intuition]

- inside a chunk : stable word order within chunk
  => secure contextual deduction at word level

- between 2 chunks : less stable chunk order within clause
  => uncertain contextual deduction at word level

1.2. Chunks & tagging words

some more features of the contextual deduction at word level

- the contextual deduction at word level relies on the chunk type :

function word tag	=> chunk type	=> content word tag
an<det>	=> noun chunk	=> an<det> order<N>
they<pron>	=> verb chunk	=> they<pron> order<V>

- tagging & chunking are easier and more accurate together
  than one after the other

1.2. Chunks & tagging words

some more features of the contextual deduction at word level

- the contextual deduction is impossible at word level
  for a chunk made of a single content word :

[the agency] [issued] [the inspection order]

there is no function word to give this chunk its type

the solution is at chunk level within the clause

Outline of the course

- 1. Standard operations in robust parsing
  - 1.1. Tagging
  - 1.2. Chunking
  - 1.3. **Clause bracketing**
- 2. Shared properties, and differences in robust parsing
- 3. Two technologies to implement symbolic rules
  - 3.1. Finite-State Transducers (FST)
  - 3.2. Engine and rules
- 4. Typical applications
- 5. Comparing robust parsing with formal grammar parsing
- 6. Introduction to the practical

1.3. Clause bracketing

tokenizing characters into words
grouping words together into chunks
grouping chunks together into clauses

it's always segmentation
at different levels

a prepositional noun chunk begins with a preposition :
<prep> => [<pN>
a subordinated clause begins with a conjunction :
<conjunction> => [<subordinated clause>

Abney 1996
Xerox - Grenoble (Aït-Moktar and Chanod 1997)
clause bracketing before chunking (Ejerhed 1996)
GREYC - Caen (Vergne, Giguet)

1.3. Clause bracketing

before chunking : a top-down idea

- clause bracketing before chunking (Ejerhed 1996)
  - an experiment on prosody of clause boundaries in read speech
- her algorithm :
  - clause segmenter --> clause "units" between 2 clause beginnings
  - clause internal parser --> complete sentence parse tree
- her conclusions :
  - clause boundaries can be recognised with great precision
    prior to any chunking
  - links between chunks within the same clause
    =/= links between chunks in 2 different clauses

Outline of the course

- 1. Standard operations in robust parsing
  - 1.1. Tagging
  - 1.2. Chunking
  - 1.3. Clause bracketing
- 2. Shared properties, and differences in robust parsing
- 3. Two technologies to implement symbolic rules
  - 3.1. Finite-State Transducers (FST)
  - 3.2. Engine and rules
- 4. Typical applications
- 5. Comparing robust parsing with formal grammar parsing
- 6. Introduction to the practical

2.1. Shared properties of robust parsing

- always a segmentation process of the input stream
- stream processing (no need to segment into sentences before parsing) and linear practical complexity
- the process is made explicit but no explicitly expected structure in input
- non recursive representations of constituent structures
  - imply a hierarchy of constituents of different types : token, chunk, clause, sentence, paragraph, ...
  - and are a "comeback" of dependency representations

2.2. Some differences inside robust parsing

- deduction process defined with statistics, or with symbolic rules
- implementations :
  - based on statistical models : Debili 1977, Church 1988, Merialdo 1991, Briscoe 1993, ...
  - based on symbolic rules
    - . finite-state transducers : Abney 1996, Ejerhed 1996, Aït-Moktar and Chanod 1997 (XRCE Grenoble), ...
    - . rules and engine : Vergne, Giguet (GREYC Caen)
- symbolic rules : reductionist deduction, or constructive deduction
- lexical resources : nearly exhaustive, or only function words

Outline of the course

- 1. Standard operations in robust parsing
  - 1.1. Tagging
  - 1.2. Chunking
  - 1.3. Clause bracketing
- 2. Shared properties, and differences in robust parsing
- 3. Two technologies to implement symbolic rules
  - 3.1. Finite-State Transducers (FST)
  - 3.2. Engine and rules
- 4. Typical applications
- 5. Comparing robust parsing with formal grammar parsing
- 6. Introduction to the practical

3. Two technologies to implement symbolic rules :

• systems using symbolic rules = Rule-Based Systems (RBS)

- 3.1. Finite-State Transducers : XRCE - Grenoble
- 3.2. Engine and rules : GREYC - Caen

• what they have in common :

- stream processing : practical linear complexity
- hand-written symbolic rules
- readable rules
- parsing : robust, less and less shallow
- a way of a renewal in parsing strategies

3.1. Finite-State Transducers (FST)

Xerox Research Centre Europe - Grenoble (XRCE)

"Regular Expressions for Language Engineering"

1 regular expression
- denotes a set of strings, or a regular language
- is represented by a **simple automaton** which recognises strings belonging to the set

2 regular expressions
- denote a set **of pairs** of strings, or a mapping between two regular languages
- are represented by a **transducer** which transduces a string of one language (input) into a string of the other language (output)

3.1. Finite-State Transducers (FST)

• Xerox new operators for regular expressions :

$A	*containment* : all strings containing a string matched by A
A => B _ C	*restriction* : A restricted in the context of B _ C
A -> B	*simple replacement* of A by B (multiple outcome)
A @-> A'	*left-to-right, longest-match replacement* (unique outcome) - may be used to mark a part of A, a NP head for instance
A @-> M1 ... M2	*left-to-right, longest-match replacement and markup* - delimits A with the markers M1 and M2 before and after - is a parser that marks maximal instances of the regular language A

NominalPhrase @-> "[NP " ... " NP]"

in replacements, the whole document is the processed string

3.1. Finite-State Transducers (FST)

• a finite-state **marker**
= a FST which introduces extra symbols into the input string

• a finite-state **filter**
= a FST which outputs only certain parts of the input string

• a finite-state light **parser** may be defined in the following way:

(1) using the longest-match and replacement operator, **FS-markers** identify noun group and verbal group boundaries

(2) **FS-markers** label the nominal or verbal heads within each group

(3) **FS-filters** extract and label the syntactic relations between words within and across group boundaries

3.1. Finite-State Transducers (FST)

• two equivalent ways to compose 2 transducers :

- compiling them into a unique transducer

- building a **cascade** of the 2 transducers :
  the output string of the first one
  is the input string to the second one

Outline of the course

3.2. Engine and rules

the GREYC parser : J. Vergne, E. Giguet, N. Lucas

• a declarative sequence of tasks

• every task uses the engine and a file of declarative rules

• algorithm of the engine for one task :

for each current unit
  for each rule
    if the rule may be applied to the current unit and its context
      then apply the rule

3.2. Engine and rules

• structure of a rule
  conditions on the current unit and its context
  => actions on the current unit and its context

• conditions : on attributes and values
  of the current unit
  of its linked units

• the context of the current unit :
  any linked unit | by a contiguity link
  | by a constituency link
  | by a functional link

3.2. Engine and rules

- structure of a rule
  conditions on the current unit and its context
  => actions on the current unit and its context
- conditions : on attributes and values
  of the current unit
  of its linked units
- actions :
  - giving a value to an attribute
  - generating a unit of the upper level
  - delivering a unit to the next task
  - linking 2 units
  - discarding a link

Outline of the course

- 1. Standard operations in robust parsing
  - 1.1. Tagging
  - 1.2. Chunking
  - 1.3. Clause bracketing
- 2. Shared properties, and differences in robust parsing
- 3. Two technologies to implement symbolic rules
  - 3.1. Finite-State Transducers (FST)
  - 3.2. Engine and rules
- 4. Typical applications
- 5. Comparing robust parsing with formal grammar parsing
- 6. Introduction to the practical

4. Typical applications

- applications where
  - stream processing,
  - constant and high rate,
  - practical linear complexity

  are a prerequisite,

… as most often in industrial contexts :

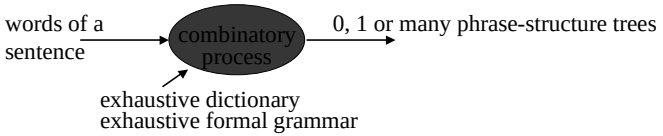
- parsing to compute prosody in a Text To Speech system
  constant rate : speech rate
- Information Retrieval on the Internet
  great amount of documents
- data, terminology, knowledge acquisition from texts

Outline of the course

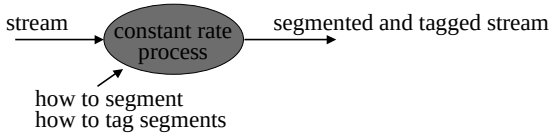
- 1. Standard operations in robust parsing
  - 1.1. Tagging
  - 1.2. Chunking
  - 1.3. Clause bracketing
- 2. Shared properties, and differences in robust parsing
- 3. Two technologies to implement symbolic rules
  - 3.1. Finite-State Transducers (FST)
  - 3.2. Engine and rules
- 4. Typical applications
- 5. Comparing robust parsing with formal grammar parsing
- 6. Introduction to the practical

5. Comparing formal grammar parsing with robust parsing

• parsing with formal grammars (HPSG, LFG, TAG, ...)



• robust parsing, partial parsing, shallow parsing



5. Comparing formal grammar parsing with robust parsing

• parsing with formal grammars (HPSG, LFG, TAG, ...)

"theoreticians" : Noam Chomsky's spiritual heirs
work on artificial material, research aim first
parsing seen as compiling : exhaustive dictionary and grammar
natural languages seen as formal languages

• robust parsing, partial parsing, shallow parsing

"empiricists" : speech recognition (HMC) spiritual heirs
work on real material, operative aim first
parsing seen as a computing process
mainly statistical methods

5. Comparing formal grammar parsing with robust parsing

• parsing with formal grammars (HPSG, LFG, TAG, ...)

recursive representation of structures :

noun phrase --> determiner + noun + phrase

• robust parsing, partial parsing, shallow parsing

non recursive representation of structures :

a chunk is made of words (but not of chunks)

5. Comparing formal grammar parsing with robust parsing

• parsing with formal grammars (HPSG, LFG, TAG, ...)

structures : expressed in an input formal grammar

process : combinatory, by exhaustive tree search

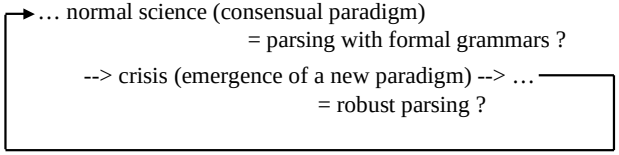
• robust parsing, partial parsing, shallow parsing

structures : computed and output

process : expressed in input contextual rules

5. Comparing formal grammar parsing with robust parsing

Thomas Kuhn's historical model :
the cycle of paradigm changes
in "The Structure of Scientific Revolutions" (1962)



Outline of the course

- 1. Standard operations in robust parsing
  - 1.1. Tagging
  - 1.2. Chunking
  - 1.3. Clause bracketing
- 2. Shared properties, and differences in robust parsing
- 3. Two technologies to implement symbolic rules
  - 3.1. Finite-State Transducers (FST)
  - 3.2. Engine and rules
- 4. Typical applications
- 5. Comparing robust parsing with formal grammar parsing
- 6. Introduction to the practical

6. Introduction to the practical

- Aim of the practical :
  - giving you a practical entrance into the field
  - bringing a concrete basis to the course
- Executing | modifying a very simple chunker | clauser
  - using the engine of the GREYC parser
  - executing a chunker for English
  - adding rules for English
  - modifying rules for French
  - linking two chunks
  - …

6. Introduction to the practical
features of the "GREYC parser"

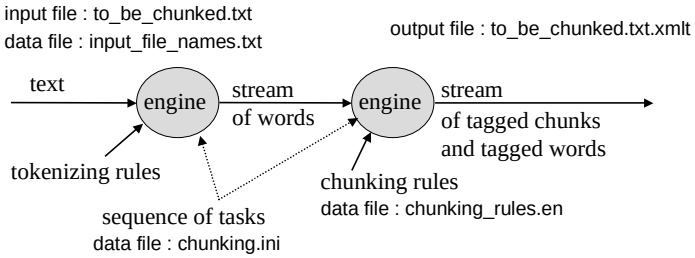
- a general platform to design and build parsers
- a generic engine (Java, 270 kb)
- a generic linguistic unit => different possible grains :
  word, chunk, clause, sentence, paragraph, ...

- declarative sequence of tasks
- every task uses the engine and a file of declarative rules
- for every task, the engine applies rules to the input stream of units

- on this platform, it is possible to build taggers, chunkers,
  "clausers", parsers, "document structurers", …
  for any language (unicode)

6. Design of a very simple chunker

using the engine of the GREYC parser



6. Design of a very simple chunker

using the engine of the GREYC parser

• Structure of the tasks sequence file :

for each detected language :
- the tasks to do
- the output task

the tasks of the simple chunker :
- splitting splits the text into paragraphs (PAR)
- tokenizing tokenizes paragraphs into words (W)
- chunking groups words together into chunks (CHK)
- output task = output of chunking

for each task : the rule files of this task, the previous task, the output unit

the constituent hierarchy is defined : Word, CHunK

6. Design of a very simple chunker

More about this constituent hierarchy :

• physical (typographical) constituents :

	the whole document
splitting	paragraphs
tokenizing	words (lowest level constituent)

• logical (computed) constituents :

chunking	chunks
clausing	clauses
sentencing	sentences

6. Design of a very simple chunker

using the engine of the GREYC parser

• Structure of the rule file :

- tagging grammatical words
  - . tagging noun chunk beginnings (prepositions, determiners)
  - . tagging verb chunk beginnings (auxiliaries)
  - . tagging chunk separators
- generating chunks
  - . generating & delivering noun chunks, verb chunk
  - . delivering chunk separators

6. Design of a very simple chunker

using the engine of the GREYC parser

• The rule formalism (designed by Emmanuel Giguet) :

tagging grammatical words :

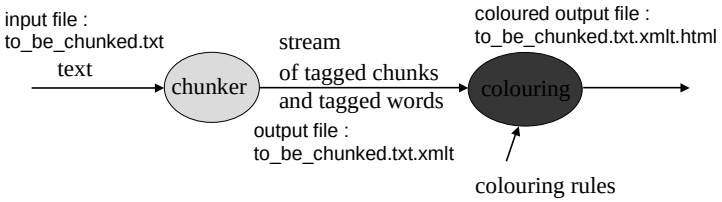
```
$0=[ G=:{ a an } ] => $0.add( [ CS=d n=s ] );
```

generating chunks :

```
$0=[ CS==p ]
   => $chk=generate( [ CS=pN ] )
     $chk.deliver();
$0=[ CS==d ] <W.next< [ CS!=p ]
   => $chk=generate( [ CS=N ] )
     $chk.deliver();
```


6. Executing a very simple chunker

using the engine of the GREYC parser



6. Also in the practical :

• Chunking English :
  - executing a very simple chunker for English
  - modifying rules

• Changing natural language :
  - making a chunker for French
  - making a chunker for another language

• Linking 2 chunks : the subject noun chunk to the verb chunk

• Changing scale of the computed unit : clause bracketing

• The genericity of the GREYC engine

the end of the course

see you this afternoon at the practical
with Emmanuel Giguet and Nadine Lucas

if you don't attend the practical,
please fill the assessment form

you will find on http://www.info.unicaen.fr/~jvergne
and during the practical
- the presentation of the course
- the practical guidelines
- references and links of the tutorial

•