Computational Construction Grammar for Open Source Collaborative Grammar Development

Remi van Trijp
Sony Computer Science Laboratory Paris
remi@csl.sony.fr

Katrien Beuls
VUB AI-Lab
katrien@ai.vub.ac.be
Tutorial Outline

• 14:00 – 15:00
  Welcome & Introduction

• 15:00 – 16:00
  Spanish verb conjugation & Error Correction

• 16:00 – 16:30
  Coffee break

• 16:30 – 18:00
  More advanced topics, discussion and conclusions

If you have questions, ask them immediately!
Tutorial Materials

• Slides
• Background reading
• Video lectures (available on 20 June)

• Fluid Construction Grammar / Babel2 software (emergent-languages.org/Babel2)
• Exercises
Introduction

Our Dream
Why Construction Grammar?
Key Ideas in Construction Grammar
Computational Construction Grammar
Collaborative Development

OPEN SOURCE GRAMMARS
Our Dream

• Coding the grammars of the world
  – Open source development of grammars
  – Create challenges and “problem sets”
  – Create a pool of solutions and design patterns
  – Set up an international network
Why use construction grammar?

- The CL/NLP field is utterly dominated by statistical processing...
- ... but there is a big elephant in the room!
The Elephant Nobody Likes to Talk About

• Evaluation studies in computational linguistics (e.g. PARSEVAL, BLEU, ...) give the impression that statistical language processing is a solved problem
• But these measures are fine-tuned to fit the model rather than evaluating how well the problem has been solved

The Elephant Nobody Likes to Talk About

• E.g. translation:
  
  *Bill schoopt de bal.* (Dutch)
  ‘Bill kicks the ball.’

• French translation (by Google translate):
  *Le projet de loi est un coup de pied de ballon.*
  Literally: The bill (i.e. law proposal) is a kick of a ball.
The Elephant Nobody Likes to Talk About

• What went wrong?
  1. All translations pass through English, even for well documented languages such as French (75M+263M speakers) and Dutch (28M speakers)
    • No semantics whatsoever
    • English is typologically a very “strange” language, hence it is not a good basis for translating to other languages
    • Mistakes are already made in this first step
    • “Bill schopt de bal.” -> “Bill is kicking the ball.”
The Elephant Nobody Likes to Talk About

• What went wrong?
  1. All translations pass through English
  2. Named entity recognition fails (even though this is considered as a solved problem):
     • Bill -> “le projet de loi”
The Elephant Nobody Likes to Talk About

• What went wrong?
  1. All translations pass through English
  2. Named entity recognition fails
  3. Failure in multiword expression
     • Correct French idiom: “donner un coup de pied à …”
     • The idiom is too long to be captured by N-gram models (in MT, 4-grams are the longest)
     • Idioms are an unsolved problem in general

The Elephant Nobody Likes to Talk About

• What went wrong?
  1. All translations pass through English
  2. Named entity recognition fails
  3. Failure in multiword expression
  4. Failure in recognizing functional structure and dependencies
    • Transitive phrases are the most frequent ones
    • Yet, “the ball” is not recognized as the direct object in the utterance, but considered as a possessive phrase that modifies the “foot”
Where is the field now?

• Statistical NLP is “crying out for better language models” (Charniak, 2001)
• “In practice (especially if we count industrial labs), a strong trend is [...] towards higher-level syntactic frameworks and hand-built grammars” (Carroll et al. 2009)
• Statistical NLP is “a pendulum swung too far” (Church 2011)
• We need to put “linguistics back in computational linguistics” (Kay 2014)
Where is the field now?

• “By ignoring [...] finer details, our language-processing systems have been stuck in an "idiot savant" stage where they can find everything but cannot understand anything. The main language processing challenge of the coming decade is to create robust, accurate, efficient methods that learn to understand the main entities and concepts discussed in any text, and the main claims made.”

  (Fernando Pereira, Google Research Director, at the META-FORUM 2012 )
Where is the field now?

• Statistics are necessary and here to stay
• But they need to be combined with more sophisticated language models
Why Construction Grammar (CxG)?

- CxG is the talk of the town in linguistics
- Computational linguists recognize the potential of its revolutionary architecture

“We suggest that the CxG perspective presents a formidable challenge to the computational linguistics/natural language processing community. [...] Systematically describing these "cross-cutting" constructions and their processing, especially in a way that scales to large data encompassing both form and meaning and accommodates both parsing and generation, would in our view make for a more comprehensive account of language processing than our field is able to offer today.” (Schneider & Tsarfaty 2013)
Introduction

CONSTRUCTION GRAMMAR
Key Ideas in Construction Grammar

• (1) Semantics, semantics, semantics!
  – Constructions are **mappings between meaning and form**
  – Hence: grammatical constructions are like lexical constructions
  – Lexicon and grammar are a continuum
e.g. *He baked a cake.*

  *He baked her a cake.*
Key Ideas in Construction Grammar

• (2) No core grammar vs. periphery of exceptions
  – EVERYTHING consists of constructions (single representation for all knowledge)
  – Of course, the linguistic inventory remains structured
Key Ideas in Construction Grammar

• (3) Constructions “cut the cake vertically”
Key Ideas in Construction Grammar

- (3) Constructions “cut the cake vertically”
Key Ideas in Construction Grammar

• (4) Constructions may freely interact with each other as long as there are no conflicts
  – Mary bought her mother a dozen roses.
  – A dozen roses, Mary bought her mother.
Constructions at work

COMPUTATIONAL CONSTRUCTION GRAMMAR
Computational Construction Grammars

• Computational construction grammar is still in an early phase of development

• 1\textsuperscript{st} International workshop on computational construction grammar will be held at the ICCG-8 Conference in Osnabrück (Germany) in September

• Currently, three more advanced projects:
  – Sign-Based Construction Grammar (SBCG)
  – Embodied Construction Grammar (ECG)
  – Fluid Construction Grammar (FCG)
Computational Construction Grammars

• Sign-Based Construction Grammar (SBCG)
  – An HPSG interpretation of construction grammar
  – Constraint-based formalism using typed feature structures
  – No computational implementation (yet)
  – Existing platforms for implementing typed feature structure grammars can be used, but no one-to-one mapping with the theory
    • Linguistic Knowledge Builder (LKB)
    • TRALE
Computational Construction Grammars

- Embodied Construction Grammar (ECG)
  - Constraint-based formalism using typed feature structures
  - Strong focus on image schemas (~ Lakovian cognitive linguistics)
  - Includes a parser, but no bidirectional processing
  - ECG Workbench & Example grammars
    [http://www1.icsi.berkeley.edu/~lucag/](http://www1.icsi.berkeley.edu/~lucag/)
Fluid Construction Grammar (FCG)

- Most advanced construction grammar implementation
  - Bidirectional processing
  - Metalayer architecture for learning
- Formalism used in this tutorial
Fluid Construction Grammar

Representing and Processing Linguistic Knowledge
Definition of a Construction
Applying Constructions
Design Patterns
Representing and Processing Linguistic Knowledge

TERM UNIFICATION
(MATCH AND MERGE)
Feature-Value Pairs

• We will represent all linguistic knowledge through pairs of features and values
• Fluid Construction Grammar uses a bracketed lisp-style notation:

$$(\text{syn-cat } ((\text{lex-class noun})
\text{ (number sg)}
\text{ (person 3)}))$$

First element is the feature-name
Second element is the feature-value
Feature-Value Pairs

• Anything can be a value
  – A symbol (person 3)
  – A list (subunits (unit-a unit-b))
  – A feature-value pair (agr (person 3))
  – A list of feature-value pairs
    (syn-cat ((lex-class noun)
      (number sg)
      (person 3)))
  – ...

Feature-Value Pairs

- Underspecified values are indicated by variables
- Variables always start with a question mark

(syn-cat ((lex-class noun)
  (number ?x)
  (person ?y)))
Why **untyped** features?

• Most grammar formalisms have adopted **typed** feature logics

• Why?
  – Theoretical background: generative grammar
    • The task of the grammar is to license all well-formed sentences, and to disallow ungrammatical ones
    • E.g. typing the verb “sneeze” as an intransitive verb prevents it from appearing in transitive clauses
  – Engineering: elegant way of capturing generalizations
Why untyped features?

• However, this approach clashes with idea of free combination of constructions
  – He sneezed.
  – He sneezed the napkin off the table.
• As a result, the grammar becomes rigid and inflexible, and this makes the job of handling novelty and mistakes in language much harder
• Instead of a generative grammar, FCG is therefore more like a transduction grammar where grammaticality is only of secondary importance
Alternative: Term Unification

• Term unification is a kind of pattern matching
  – Is pattern A equal to pattern B?
    e.g. (unify ‘a ‘a) => yes
        (unify ‘a ‘b) => no
  – When there are variables, the question becomes: can I find a substitution for the variables so that two patterns become equal?
    e.g. (unify ‘?x ‘a) => yes, because ?x can be bound to the value a
Hands-on exercises

• We will now learn to understand term unification through exercises, focusing on:
  – “matching” (term unification)
  – “merging” (combining two patterns using variable bindings obtained through matching)

• If you do not have FCG installed yet, please sit together with someone who has
Towards construction grammar

DEFINITION OF A CONSTRUCTION
What is a construction?

- Informal definition: mapping between meaning and form
- FCG definition: **coupled** feature structure
  - A semantic pole
    (including meaning, semantic categorization, pragmatics, ...)
  - A syntactic pole
    (including syntax, morphology, phonology, ...)
What is a construction?

• Some concerns:
  – Typological diversity: the formalism should be general and avoid making linguistic choices (e.g. some languages have no constituent structure!)
  – Key idea in CxG: constructions must be able to reach all information, no matter how deep they might be embedded in the linguistic structure
def-construction

\[(\text{def-construction } \text{name} \ (\text{attributes}) \ \text{semantic-pole} \ \longleftrightarrow \ \text{syntactic-pole})\]

- As we will see later, the distinction between two poles will allow us to process the construction for both parsing and production using the same processing algorithm
Unit structure

• In order to address the concern of linguistic diversity, we dissociate the formal representation from the linguistic structure through **units**
  
  (unit-name
    unit-body)

• A unit is like a box that collects feature-value pairs that belong together:
  
  (tutorial-word
    (syn-cat ((lex-class noun)
      (person 3)
      (number sg))))
Unit structure

• If a language has e.g. constituency structure, it is explicitly declared in terms of feature-value pairs as well

(noun-phrase-unit
 (syn-subunits (determiner noun))
 (syn-cat ((phrase-type-type NP)
     (number sg)
     (person 3)))))
Coupled feature structures

- The semantic and syntactic pole are basically a flat list of units, which each carry feature-value pairs:

  ((unit-1
      unit-body-1)
   (unit-2
      unit-body-2)
   ...
   (unit-n
      unit-body-n))
Hands-On Exercise

- We will now look at the definition of the most simple construction possible, consisting of a meaning and a form
  - Meaning: in this tutorial, we use a simple first-order predicate calculus for representing meaning using a prefix notation, e.g. $book(\ ?x)$ becomes: $(book \ ?x)$
  - Form: in this tutorial, we will also use a first-order predicate calculus for describing the form of words, e.g. $(string \ ?unit \ \text{"book"})$
  - Summary: both meaning and form are features whose value consist of a list of logic predicates

- We will also look at the graphical representation of the construction in our web interface
Transient Structures

APPLYING CONSTRUCTIONS
Transient Structure

• When parsing or producing an utterance, FCG keeps all information obtained about that utterance in a **transient structure**
• A transient structure is, just like a construction, a mapping between a semantic and syntactic pole
• Transient structures start simple but grow more elaborate as constructions add more information to them during processing
Applying a Construction

transient structure

semantic pole

syntactic pole

construction

semantic pole

syntactic pole

first merging phase

matching phase

second merging phase

second merging phase

matching phase

first merging phase
Applying a construction

• We will now do the exercises in lrec02.lisp
Common Solutions

DESIGN PATTERNS