GF2UD and UD2GF

UD: *Universal Dependencies*

Prasanth Kolachina
GF Summer school, 2017
the black cat sees us today

le chat noir nous voit aujourd’hui

dependency parser

ud2gf

gf2ud

GF
Universal Dependencies
Principles of Design

- UD needs to be satisfactory on linguistic analysis grounds for individual languages.
- UD needs to be good for linguistic typology, i.e., providing a suitable basis for bringing out cross-linguistic parallelism across languages and language families.
- UD must be suitable for rapid, consistent annotation by a human annotator.
- UD must be suitable for computer parsing with high accuracy.
- UD must be easily comprehended and used by a non-linguist .... (API grammar)
- UD must support well downstream language understanding tasks (relation extraction, reading comprehension, machine translation, ...).
Mission of Grammatical Framework

The mission of GF is to formalize
the grammars of the world

and

make them available for computer applications.
Universal Dependencies

A community-driven effort to annotate multilingual treebanks

Cross-lingual consistency in annotations across languages

17 Part-of-Speech tags ; 40 dependency labels ; morphological features

Annotated corpora released every 6 months;

Ongoing V2

50 Languages, 70 Treebanks
the black cat sees us today
Clausal Predicates

nsubj
dobj iobj

Copulas
cop

Noun dependents
det amod

Auxiliary verbs and negation
aux neg
Structures in GF
the     black       cat         sees
         us
PredVP

DetCN
the_Det black_AP

AdjCN
cat_CN

ComplTV

see_TV

UsePron
we_Pron
the black cat sees us
S
  / \
 /   \\
NP   VP
  /   /
 /    /
Det  CN  NP  TV
  /    /
 /     /
CN  AP  Pron
  /    /
 /     /
le  chat  noir  nous  voit
<table>
<thead>
<tr>
<th></th>
<th>dependencies</th>
<th>GF</th>
</tr>
</thead>
<tbody>
<tr>
<td>parsing robustness</td>
<td>robust</td>
<td>brittle</td>
</tr>
<tr>
<td>parsing speed</td>
<td>fast</td>
<td>slow</td>
</tr>
<tr>
<td>semantics</td>
<td>loose</td>
<td>compositional</td>
</tr>
<tr>
<td>generation</td>
<td>?</td>
<td>accurate</td>
</tr>
</tbody>
</table>
the black cat sees us today

dependency parser

le chat noir nous voit aujourd’hui

dependency parser

ud2gf

gf2ud

GF
the black cat sees us today

∃ !A. (cat(A) & MODIFIER(black,A) &
(∃ B. (see(B) & SUBJECT(B)=A & OBJECT(B) = we & MODIFIER(today,B))))
GF2UD

grammatical roles to arguments and hide functions
Dependency configuration
PredVP  nsubj head
ComplTV  head  dobj
DetCN  det  head
AdjCN  amod  head

PredVP
  /\                 /\  \
DetCN  ComplTV  the_Det  AdjCN  see_TV  UsePron
  /\                   /\       /\       /
black_AP  cat_CN  we_Pron
Dependency configuration
PredVP  nsubj head
ComplTV  head  dobj
DetCN  det  head
AdjCN  amod  head

det

nsubj

amod

PredVP

Dobj

DetCN

ComplTV

UsePron

the_Det

AdjCN

see_TV

black_AP

cat_CN

we_Pron
The black cat saw the TV.
The black cat sees us.
the black cat sees us
Le chat noir nous voit.
le chat noir nous voit
Syncategorematic words

- pinpointing a difference in the ways of thinking:
  - dependency grammar is about words,
  - GF is about meanings
the cat is black
categorematic: word with its own category and function

fun cat_CN : CN
lin cat_CN = “cat”

syncategorematic: word that is “between categories”

fun ComplAP : AP -> VP
lin ComplAP ap = “is” ++ AP

No semantics (fun) of its own. Not an argument. No label.
the cat is black
adding default labels
we get

UD wants
Other syncategorematic words

- negation words
- tense auxiliaries
- infinitive marks
- (sometimes) prepositions
Extended dependency configuration

abstract | concrete  local | nonlocal

- more complicated, not universal
+ less work than rewriting the grammar anyway
+ UD is still undergoing changes
Concrete configs

UseComp in English

UseComp head {“is”, “was”, “be”, “are”} cop head

In Swedish

UseComp head {“ar”, “var”, “vara”, “varit”} cop head
Local Concrete configurations

Mappings defined on linearization of an abstract function for a specific language

These are necessary because of the "level of abstraction" in GF abstract syntax

The mappings specify re-labelling operations

  relabel an existing edge with new label

  modify an existing edge by changing the head and adding a new label

These operations match a set of words, or a record field or match anything
Demo ?
<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>the_Det</th>
<th>DET</th>
<th>Det</th>
<th>2</th>
<th>det</th>
<th>_</th>
<th>_</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>cat</td>
<td>cat_CN</td>
<td>NOUN</td>
<td>CN</td>
<td>3</td>
<td>nsubj</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>3</td>
<td>sees</td>
<td>see_TV</td>
<td>VERB</td>
<td>TV</td>
<td>0</td>
<td>dep</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>4</td>
<td>us</td>
<td>we_Pron</td>
<td>PRON</td>
<td>Pron</td>
<td>3</td>
<td>dobj</td>
<td>_</td>
<td>_</td>
</tr>
</tbody>
</table>
The black cat sees us today.
the black cat sees us today

1 the the DET _ 3 det
2 black black ADJ _ 3 amod
3 cat cat NOUN _ 4 nsubj
4 sees see VERB _ 0 root
5 us we PRON _ 4 dobj
6 today today ADV _ 4 advmod
the black cat sees us today

tree

root see VERB _ 4

  nsubj cat NOUN _ 3

    det the DET _ 1

      amod black ADJ _ 2

  dobj we PRON _ 5

    advmod today ADV _ 6
the black cat sees us today
The black cat sees us today.
Postorder traversal: subtrees before their head

Invariant: every node has a valid GF tree

Goal: total GF tree at root
A node is done when no more functions apply
when an endocentric function applies, use it first
tree

root see_V2 V2 4

  nsubj (DetCN 1 3) [(ModCN 2 3),(UseN 3),cat_N] NP 3

    det the_Det Det 1

    amod (PositA 2) [black_A] AP 2

  dobj (UsePron 5) [we_Pron] NP 5

    advmod today_Adv Adv 6
Root node contains a complete GF tree
Problems

Ambiguity
There can be several candidate Functions and Categories.

Incompleteness
The tree may have nodes not referenced from the AST.
Problems and solutions

Ambiguity
There can be several candidate Functions and Categories.

Maintain a list of trees at each node, not just one tree.

Incompleteness
The tree may have nodes not referenced from the AST.

Auxiliary rules for syntcategorematic words.

Backup functions attached as adverbial modifiers to AST nodes.
Fast and friendly service, they know my order when I walk in the door!
if a man owns a donkey it beats he
if if SCONJ SCONJ _ 4 mark _ _
a a DET DET Definite=Ind|PronType=Art 3 det _ _
man man NOUN NOUN Number=Sing 4 nsubj _ _
owns own VERB VERB Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin 8 advcl _ _
a a DET DET Definite=Ind|PronType=Art 6 det _ _
donkey donkey NOUN NOUN Number=Sing 4 dobj _ _
it it PRON PRON Case=Nom|Gender=Neut|Number=Sing|Person=3|PronType=Prs 8 nsubj _ _
beats beat VERB VERB Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin 0 root _ _
he he PRON PRON Case=Nom|Gender=Masc|Number=Sing|Person=3|PronType=Prs 8 dobj _ _
Experiments

Analysing and converting UD treebanks
  - English, Finnish, Swedish

Connecting GF generation to UD parser front-end
## Results

<table>
<thead>
<tr>
<th>language</th>
<th>#trees</th>
<th>#confs</th>
<th>%cov’d</th>
<th>%int’d</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>2077</td>
<td>45</td>
<td>98</td>
<td>75</td>
</tr>
<tr>
<td>Finnish</td>
<td>648</td>
<td>20</td>
<td>91</td>
<td>60</td>
</tr>
<tr>
<td>Finnish*</td>
<td>648</td>
<td>0</td>
<td>81</td>
<td>59</td>
</tr>
<tr>
<td>Swedish</td>
<td>1219</td>
<td>35</td>
<td>94</td>
<td>68</td>
</tr>
<tr>
<td>Swedish*</td>
<td>1219</td>
<td>0</td>
<td>84</td>
<td>63</td>
</tr>
</tbody>
</table>

UD_English/en-ud-test.conllu

UD_Finnish/fi-ud-test.conllu

UD_Swedish/sv-ud-test.conllu
Demo and End-user Applications
> parse “the cat sees us” | visual_dep -output=conll -file=ud.labels

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>the_Det</td>
<td>DET</td>
<td>Det</td>
<td>_</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>cat_CN</td>
<td>NOUN</td>
<td>CN</td>
<td>_</td>
</tr>
<tr>
<td>3</td>
<td>sees</td>
<td>see_TV</td>
<td>VERB</td>
<td>TV</td>
<td>_</td>
</tr>
<tr>
<td>4</td>
<td>us</td>
<td>we_Pron</td>
<td>PRON</td>
<td>Pron</td>
<td>_</td>
</tr>
</tbody>
</table>

2 det _ _
3 nsubj _ _
0 root _ _
3 dobj _ _
UD2GF

$ ud2gf

-lEng -t10000 -k3000 -a1 -g1 -Dscamifgtn
-CUDTranslate.labels,UDTranslateEng.labels
treebanks/UD_English/en-ud-test.conllu

https://github.com/GrammaticalFramework/gf-contrib/tree/master/ud2gf
UD pipelines

SyntaxNet : Google’s parser -- not lemmatizer

Stanford CORENLP -- no morphological analysis

UDPipe

Inhouse Graph-based Parsing pipeline
Questions ?
Non-local Abstract mappings

Expression patterns correspond to sub-trees or multi-level rules in GF Abstract Syntax

Have higher precedence that the corresponding local rule for the top function

But, we could get rid of these non-local mappings by re-engineering the RGL Abstract Syntax quite easily

Could result in an increase of grammar size
Non-local Abstract mappings

(PredVP ? (PassV2 ?)) nsubjpass head

PredVP nsubj head

(PredSCVP ? (PassV2 ?)) csubjpass head

PredSCVP csubj head
Some sources of non-universal configurations

ComplV2 : V2 -> NP -> VP    head (dobj | iobj | nmod)
ComplVV : VV -> VP -> VP    aux head | head xcomp
                      | head mark xcomp
ExistNP : NP -> Cl        “there is”, “det finns”,...
Non-Local Concrete mappings

Syncategoramatic words introduced by non-local rule

The same expression patterns from Non-local Abstract rules are used, to specify relabelling operations

(UseCl ? PNeg ?) head {“not”, “n’t”} neg head
(UseCl ? ? ?) head {*} aux head

Auxiliaries for passive voice constructions?

(UseCl ? ? (PredVP ? (PassV2 ?))) auxpass head