

Heart of Gold Intro & Update

Ulrich Schäfer

DELPH-IN Summit 2007, Berlin
August 21, 2007

Idea: Combine Deep & Shallow NLP Components

Synergy, division of labor:

Shallow NLP

- Open class words (e.g. named entities)
- Local information
- Domain-specific modelling

Deep NLP

- Exhaustive, non-local analysis
 - Long distance dependencies
 - Negation scope
 - Predicate-argument structure
- Semantics representation

→ Increased robustness of deep parsing

Why Architecture?

- Build on many pre-existing tools
- Abstraction from specific components
- Standards for linguistic annotation are only emerging
- Mediation between namings and structures (→ XSLT)
- Provide online services for applications such as QA (as opposed to offline, corpus-based integration)

Why XSLT?

XSLT

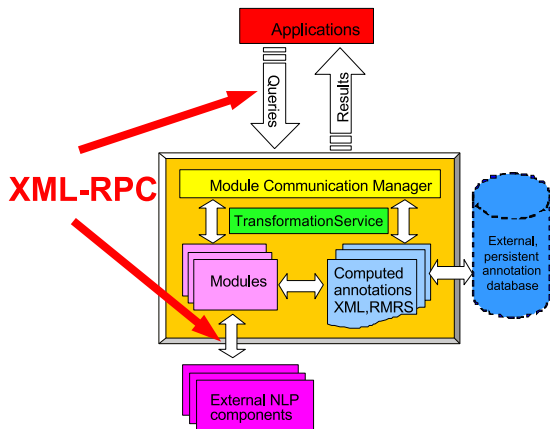
- XML tree transformation language
- Mature W3C recommendation
- XPath as sub-language
- Fast, portable implementations

Motivation

- Mappings between different representation formats
- XPath alone not powerful enough
- Corpus query languages too slow & specialized

Heart of Gold - Design Principles

- Application-oriented middleware architecture
- Optional common semantics representation format RMRS [Copestake 2003]
- Fallback to shallow result in case deep parsing fails
- Open to other XML annotation formats
- Networkability
- Multilinguality
- Configurability, flexibility

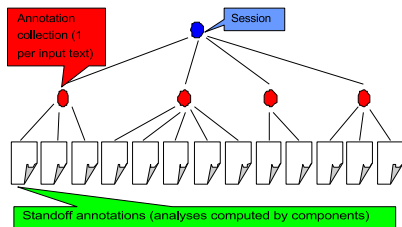


Integrated Components

Component	NLP Type	Languages	Implemented in
JTok	tokenizer	de, en, it,...	Java
ChaSen	Japanese morph.	ja	C
TnT	statistical tagger	de, en,...	C
Treetagger	statistical tagger	en, de, es, it,...	C
Chunkie	stat. chunker	de, en,...	C
ChunkieRMRS	chunk RMRSes	de, en	XSLT, SDL/Java
LingPipe	statistical NER	en, es,...	Java
Sleepy	shallow parser	de	OCaml
SProUT	shallow NLP/NER	de, el, en, ja,...	Java
LoPar/wbtopo	PCFG parser	de	C, XSLT
Corcy	coref resolver	en	Python
RASP	shallow NLP	en	C, Lisp
PET	HPSG parser	de, el, en, ja,...	C, C++, Lisp
RMRSmerge	RMRS merger	de, en,...	XSLT, SDL/Java
SDL	sub-architectures		SDL/Java

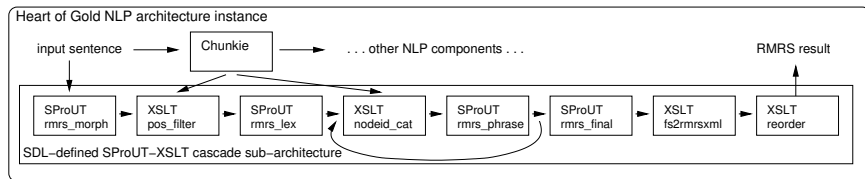
Code Generation 1: SDL [Krieger 2003]

- Idea: compile declaratively defined sub-architectures to Java code
- Original motivation: cascaded SProUT grammars with sequence, parallelism, mediators, loops
- HoG SdlModule instance encapsulates sub-architecture
- Generic SProUT and XSLT modules defined
- Access to other annotations via XPath URI `hog://sid/acid/aid`



Implemented SDL Sub-architectures

- RMRMerge combines shallow (e.g. NER) and deep RMRsEs: 5x XSLT, language-independent and module-independent
- ChunkieRMRS [Frank et al 2004]: 4x SProUT, 4x XSLT, + input from Chunkie via HoG URI (en + de)



Code Generation 2: XSLT Mappings

- Automatic XSLT code generation from declaratively specified type interfaces in SProUT
- Input: SProUT structured output type definition
- Output: XSLT stylesheet for online transformation of SProUTput into RMRS and PET XML input chart format

```
[ ne-person
  CSTART      "4"
  CEND        "23"
  P-POSITION  CEO
  TITLE       <"Dr.">
  GIVEN_NAME  <"Peter", "Paul">
  SURNAME     [1] "Wirth"
  VARIANT     [1] "Wirth"
]
```

→

```
<pet-input-chart>
  <w prio="0.5" constant="yes" cend="23" cstart="0" id="SPR1">
    <surface>CEO Dr. Peter Paul Wirth</surface>
    <typeinfo baseform="no" id="TIN1">
      <stem>$genericname</stem>
    </typeinfo>
  </w>
</pet-input-chart>
```

↓

```
[ TEXT [1] CEO Dr. Peter Paul Wirth
  TOP h100
  RELS {
    [ ne-person_rel ] [ variant_rel ] [ p-position_rel ] [ surname_rel ] [ given_name_rel ] [ title_rel ]
    LBL h100          LBL h110          LBL h111          LBL h112          LBL h113          LBL h114
    ARG0 x100         ARG0 x110         ARG0 x111         ARG0 x112         ARG0 x113         ARG0 x114
    CARG [1]          CARG Wirth        CARG CEO          CARG Wirth        CARG Peter Paul   CARG Dr.
    ARG1 x100         ARG1 x100         ARG1 x100         ARG1 x100         ARG1 x100         ARG1 x100
  }
```

Deep Parser Integration: PET XML Input Chart

- XML format generated from PoS tagger (TnT) and NER (SProUT) via XSLT

```
<pet-input-chart>
  <w id="TNT0" cstart="0" cend="3">
    <surface>When</surface>
    <pos tag="WRB" prio="1.000000e+00"/>
  </w>
  <w id="TNT1" cstart="5" cend="7">
    <surface>was</surface>
    <pos tag="VBD" prio="1.000000e+00"/>
  </w>
  <w id="TNT2" cstart="9" cend="15">
    <surface>Algeria</surface>
    <pos tag="NNP" prio="1.000000e+00"/>
  </w>
  <w id="TNT3" cstart="17" cend="25">
    <surface>colonized</surface>
    <pos tag="VBD" prio="7.928154e-01"/>
    <pos tag="VBN" prio="2.071846e-01"/>
  </w>
  <w id="TNT4" cstart="26" cend="26" constant="yes">
    <surface>?</surface>
    <pos tag="?" prio="1.0"/>
  </w>
  <w id="SPR1" const="yes" cend="15" cstart="9" prio="0.5">
    <surface>Algeria</surface>
    <typeinfo baseform="no" id="TIN2.1">
      <stem>$genericname</stem>
    </typeinfo>
  </w>
</pet-input-chart>
```

Deep Parser Integration: PET XML Input Chart

- XML format generated from **PoS tagger (TnT)** and **NER (SProUT)** via **XSLT**

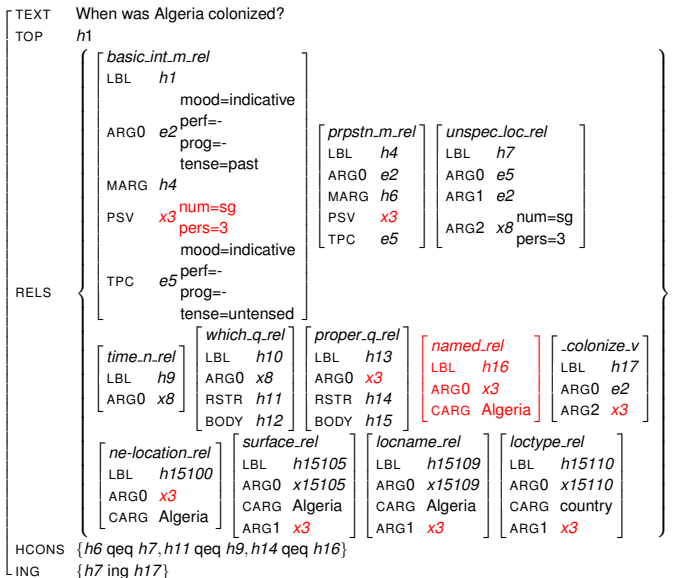
```
<pet-input-chart>
  <w id="TNT0" cstart="0" cend="3">
    <surface>When</surface>
    <pos tag="WRB" prio="1.000000e+00"/>
  </w>
  <w id="TNT1" cstart="5" cend="7">
    <surface>was</surface>
    <pos tag="VBD" prio="1.000000e+00"/>
  </w>
  <w id="TNT2" cstart="9" cend="15">
    <surface>Algeria</surface>
    <pos tag="NNP" prio="1.000000e+00"/>
  </w>
  <w id="TNT3" cstart="17" cend="25">
    <surface>colonized</surface>
    <pos tag="VBD" prio="7.928154e-01"/>
    <pos tag="VBN" prio="2.071846e-01"/>
  </w>
  <w id="TNT4" cstart="26" cend="26" constant="yes">
    <surface>?</surface>
    <pos tag="?" prio="1.0"/>
  </w>
  <w id="SPR1" const="yes" cend="15" cstart="9" prio="0.5">
    <surface>Algeria</surface>
    <typeinfo baseform="no" id="TIN2.1">
      <stem>$genericname</stem>
    </typeinfo>
  </w>
</pet-input-chart>
```

Deep Parser Integration: PET XML Input Chart

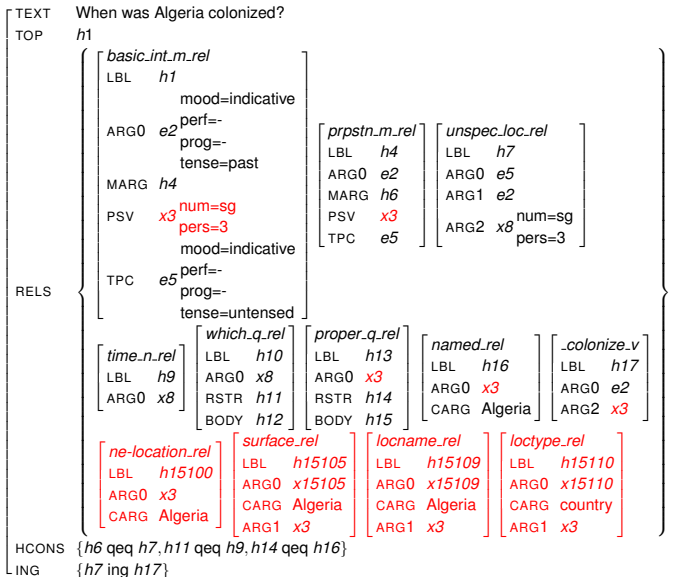
- XML format generated from PoS tagger (TnT) and NER (SProUT) via XSLT

```
<pet-input-chart>
  <w id="TNT0" cstart="0" cend="3">
    <surface>When</surface>
    <pos tag="WRB" prio="1.000000e+00"/>
  </w>
  <w id="TNT1" cstart="5" cend="7">
    <surface>was</surface>
    <pos tag="VBD" prio="1.000000e+00"/>
  </w>
  <w id="TNT2" cstart="9" cend="15">
    <surface>Algeria</surface>
    <pos tag="NNP" prio="1.000000e+00"/>
  </w>
  <w id="TNT3" cstart="17" cend="25">
    <surface>colonized</surface>
    <pos tag="VBD" prio="7.928154e-01"/>
    <pos tag="VBN" prio="2.071846e-01"/>
  </w>
  <w id="TNT4" cstart="26" cend="26" constant="yes">
    <surface>?</surface>
    <pos tag="?" prio="1.0"/>
  </w>
  <w id="SPR1" const="yes" cend="15" cstart="9" prio="0.5">
    <surface>Algeria</surface>
    <typeinfo baseform="no" id="TIN2.1">
      <stem>$genericname</stem>
    </typeinfo>
  </w>
</pet-input-chart>
```

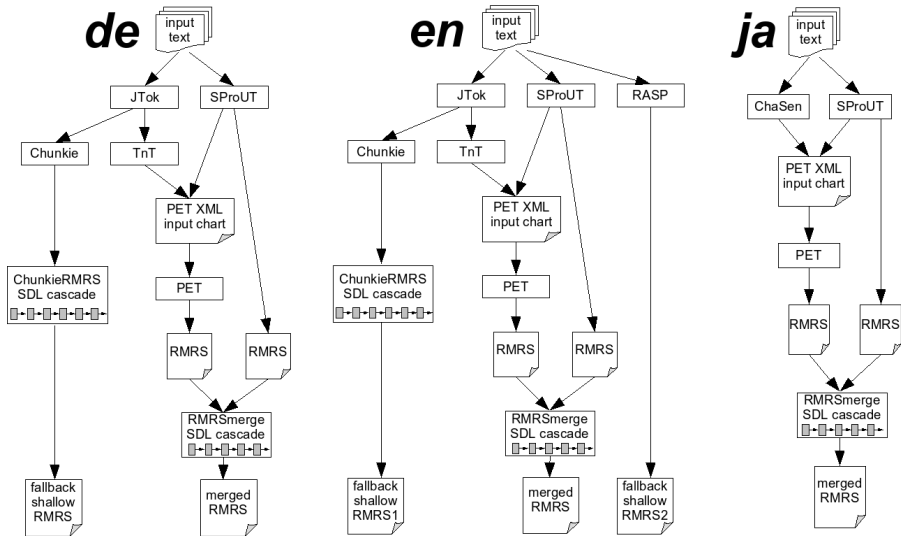

Merged RMRS (PET + SProUT)



Merged RMRS (PET + SProUT)



Hybrid Workflows



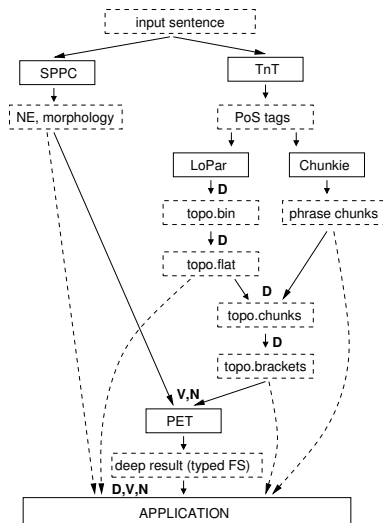
- New German Topoparser Module based on LoPar (WHITEBOARD)
- New (preliminary) Module for FreeLing
- Improved ChaSen Integration (+SProUT for Japanese)
- (Potentially) faster XSLT: Xalan-J with translet compilation
- Generic (XSLT-based) and Module-specific SMAF support added
- Module updates: Sprout, PET, LingPipe
- Problematic components (as always): PET, RASP

Shallow Topoparser Integration

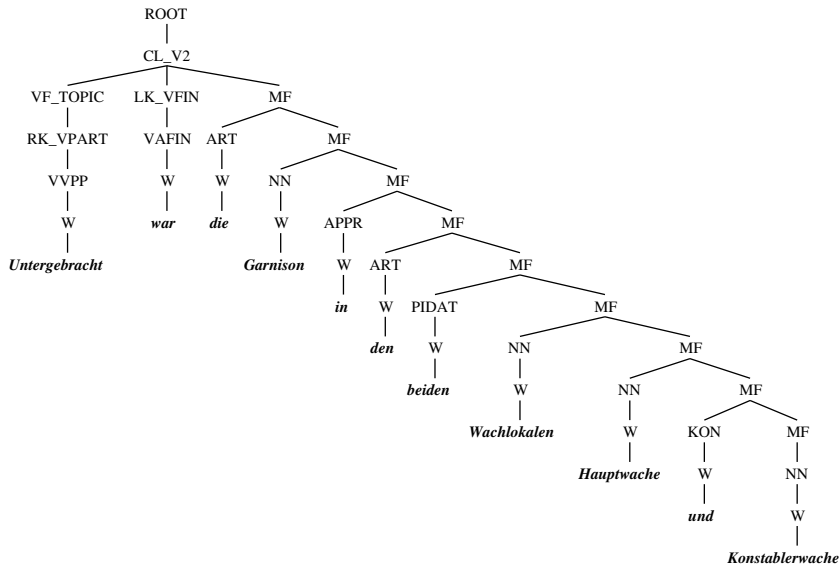
6 shallow NLP levels integrated
[Frank et al 2003]:

- PoS, Morph, NER for improved lexical robustness
- Tok, PoS, Chunk, Topo to shape search space of deep parser
- currently, PET in Heart of Gold is not yet using topo information (WHITEBOARD extensions will have to be re-integrated)

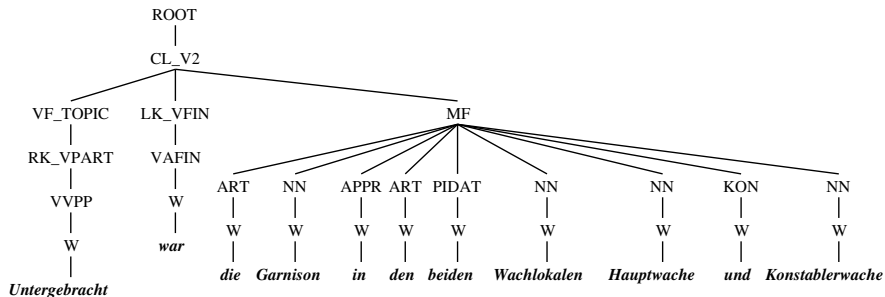
(D): XSLT for topotree transformations (sentence-wise)



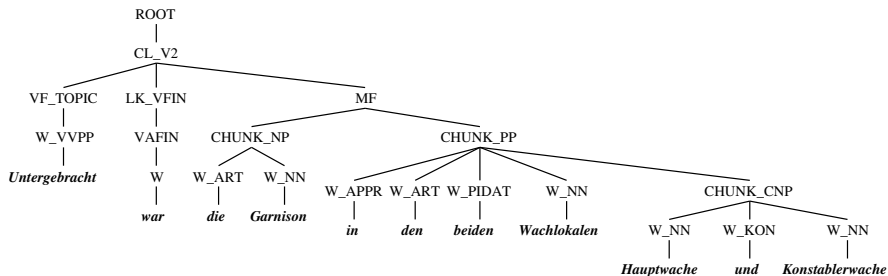
Topoparser Transformation 1: Binary Tree



Topoparser Transformation 2: Flatten Tree



Topoparser Transformation 3: Chunks



- move to public SVN + trac on **heartofgold.opendfki.de**
- automatic testing
- integrate new grammar versions?