Surfacing Dutch syntactic parses

Erwin R. Komen
Erwin_komen@sil.org
CLIN-26
Amsterdam
December 18, 2015
Pre-amble

What is surfacing??
What is surfacing??

1. Reaching the surface level?  $\rightarrow$ no…
Pre-amble

What is surfacing??

1. Reaching the surface level? → no…
2. Making a surface flat? → no…
Pre-amble

What is surfacing??

1. Reaching the surface level? \(\rightarrow \) no…
2. Making a surface flat? \(\rightarrow \) no…
3. Restoring ‘surface’ word order structure in a parsed sentence
Pre-amble

What is surfacing??

1. Reaching the surface level? \(\rightarrow\) no…
2. Making a surface flat? \(\rightarrow\) no…
3. Restoring ‘surface’ word order structure in a parsed sentence
Pre-amble

What is surfacing??

1. Reaching the surface level? → no…
2. Making a surface flat? → no…
3. Restoring ‘surface’ word order structure in a parsed sentence
Pre-amble

What is surfacing??

1. Reaching the surface level? → no…
2. Making a surface flat? → no…
3. Restoring ‘surface’ word order structure in a parsed sentence
   a) Splitting discontinuous-constituents

---

Royal Netherlands Academy of Arts and Sciences
Pre-amble

What is surfacing??

1. Reaching the surface level? → no…
2. Making a surface flat? → no…
3. Restoring ‘surface’ word order structure in a parsed sentence
   a) Splitting discontinuous-constituents
Contents

1. Why ‘surfacing’?
2. Our goal
3. The algorithm
4. Implementation
5. Discussion
Contents

1. Why ‘surfacing’?
2. Our goal
3. The algorithm
4. Implementation
5. Discussion
1 – Why surfacing?

1. Linguists need surface word order
   a. Scrambling

De universiteit stelde hem aan als hoogleraar

De universiteit stelde hem als hoogleraar aan
1 – Why surfacing?

1. Linguists need surface word order
   a. Scrambling
   b. Extraposition

De vogels die er zwart invallen zijn het onderwerp

Hij wijst hem de vogels aan die er zwart invallen
1 – Why surfacing?

1. Linguists need surface word order
   a. Scrambling
   b. Extraposition
   c. Information structure

Op het toneel verscheen een kleine man met een rugzak

Een kleine man met een rugzak verscheen op het toneel
1 – Why surfacing?

1. Linguists need surface word order
   a. Scrambling
   b. Extraposition
   c. Information structure
   d. First-constituent research

Natuurlijk gaan we op zomervakantie

We gaan natuurlijk op zomervakantie

Op zomervakantie gaan we natuurlijk

First constituent
1 – Why surfacing?

1. Linguists need surface word order
   a. Scrambling
   b. Extraposition
   c. Information structure
   d. First-constituent research
   e. Verb-cluster order

*Ik zou dat gezien willen hebben*

*Ik zou dat willen hebben gezien*
1 – Why surfacing?

1. Linguists need surface word order
   a. Scrambling
   b. Extraposition
   c. Information structure
   d. First-constituent research
   e. Verb-cluster order

2. Dependency structure sometimes obscures surface order
   a. Dependency $\rightarrow$ nice constituents
   b. Alpino: attributes for surface order
   c. Xpath/Xquery searches: huge challenge for linguists
1 – Why surfacing?

Dependency structure obscures surface order:
1 – Why surfacing?

Dependency structure obscures surface order:
1 – Why surfacing?

Dependency structure obscures surface order:
1 – Why surfacing?

Dependency structure obscures surface order:
1 – Why surfacing?

1. Linguists need surface word order
   a. Scrambling
   b. Extraposition
   c. Information structure
   d. First-constituent research
   e. Verb-cluster order

2. Dependency structure obscures surface order
   a. Dependency → nice constituents
   b. Alpino: attributes for surface order
   c. Xpath/Xquery searches: huge challenge for linguists

3. Surfacing retains the dependency information
   a. Constituent-parse: surface order
   b. Split-constituents: retrievable
Contents

1. Why ‘surfacing’?
2. Our goal
3. The algorithm
4. Implementation
5. Discussion
2 – Goal

1. Transform into surface-continuous constituents
   a. Split discontinuous constituents
2 – Goal

1. Transform into *surface-continuous constituents*
   a. Split discontinuous constituents
   b. Correct naming of parts
2 – Goal

1. Transform into **surface-continuous** constituents
   a. Split discontinuous constituents
   b. Correct naming of parts
   c. Re-order parts

Royal Netherlands Academy of Arts and Sciences
2 – Goal

1. Transform into surface-continuous constituents
   a. Split discontinuous constituents
   b. Correct naming of parts
   c. Re-order parts
   d. Re-locate displaced parts
2 – Goal

1. Transform into *surface-continuous* constituents
   a. Split discontinuous constituents
   b. Correct naming of parts
   c. Re-order parts
   d. Re-locate displaced parts
   e. Leave markers in re-location and source
2 – Goal

1. Transform into **uninterrupted** constituents
   a. Split discontinuous constituents
   b. Correct naming of parts
   c. Re-order parts
   d. Re-locate displaced parts
   e. Leave markers in re-location sources

2. Retain **dependency** constituent information
   a. Referential link between re-location and source
   b. Re-located constituent: POS-\( n \)
   c. Source constituent: *ICH*-\( n \) child
      *(Interpret Constituent Here)*
      - Used successfully in historical English corpora
      - Allows retrieving the dependency constituents
Contents

1. Why ‘surfacing’?

2. Our goal

3. The algorithm

4. Implementation

5. Discussion
3 – The algorithm

Bottom-up

in eerste instantie wordt een dienstverband van 2 jaar geboden .

1 2 3 4 5 6 7 8 9 10 11
3 – The algorithm

Bottom-up

The algorithm

in eerste instantie wordt een dienstverband van 2 jaar geboden.
3 – The algorithm

Bottom-up
a. Copy + sort words
3 – The algorithm

Bottom-up

a. Copy + sort words
b. Treat word #1
3 – The algorithm

Bottom-up

a. Copy + sort words

b. Treat word #1, #2
   #2: target parent [pp-mod]
3 – The algorithm

Bottom-up

a. Copy + sort words
b. Treat word #1, #2

#2: target parent [pp-mod]
Chk: 1st word after target may not precede me
3 – The algorithm

Bottom-up

a. Copy + sort words
b. Treat word #1 … #4
3 – The algorithm

Bottom-up

a. Copy + sort words

b. Treat word #1 … #9
3 – The algorithm

Bottom-up

a. Copy + sort words, treat #1…#9
b. Word #10: under [ppart-vc]?
   a. Chk: 1st word after target may not precede me…
   b. Placeholder ICH-1
   c. Put [ww-1] ‘lowest common ancestor’
3 – The algorithm

Bottom-up

a. Copy + sort words, treat #1…#9

b. Word #10: under [ppart-vc]?
   a. Chk: 1st word after target precedes me…
   b. Placeholder ICH-1
   c. Put [ww-1] ‘lowest common ancestor’ of:
      • ICH-1
      • ‘jaar’
3 – The algorithm

Bottom-up

a. Copy + sort words
b. Treat word #1 … #9
c. Word #10: under [ppart-vc]?
   a. Placeholder ICH-1
   b. Put [ww] ‘lowest common ancestor’
d. Trace ‘obj1’ → ‘np-su’
3 – The algorithm

Bottom-up: summary

1. Copy and sort words: source $\rightarrow$ destination
2. Mark non-words as “not done” in source
3. Treat each word in source:
   i. Consider each constituent above source.word (until but not including the top)
   ii. IF constituent has not been done yet
      i. Insert constituent above
      ii. Remember location of constituent in destination
   iii. ELSE
      i. target = corresponding parent of constituent in destination
      ii. IF “1st word after ‘target’” does not precede word”
         i. Append constituent under target
      iii. ELSE
         i. Append *ICH*-n placeholder under target
         ii. Append constituent under common ancestor of
            i. target and
            ii. last word in destination that precedes word
4. Treat all ‘traces’ (constituents not dominating a word)
5. Treat all punctuation
Contents

1. Why ‘surfacing’?
2. Our goal
3. The algorithm
4. Implementation
5. Discussion
4 – The implementation

1. Alpino to Psdx conversion
   a. Part of “Cesax” → http://erwinkomen.ruhosting.nl
4 – The implementation

1. Alpino to Psdx conversion
   a. Part of “Cesax” → http://erwinkomen.ruhosting.nl

2. Alpino to FoLiA conversion
   a. Stand-alone C#
      • Run by ‘mono’ on Windows, Mac, Linux
      • Github → http://github.com/ErwinKomen/FoliaParse
   b. Alpino: Sonar500 part of Lassy-Groot
   c. FoLiA: standard for ‘Nederlab’
   d. Performance:

<table>
<thead>
<tr>
<th>Part</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section</td>
<td>WR-P-E-C_e-magazines</td>
</tr>
<tr>
<td>Files</td>
<td>18875</td>
</tr>
<tr>
<td>Size tagged FoLiA</td>
<td>3.2 Gb</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel Xeon 1.80 GHz, 128 Gb</td>
</tr>
<tr>
<td>Time</td>
<td>6 h, 24 m</td>
</tr>
</tbody>
</table>
4 – The implementation

Resulting trees:
4 – The implementation

Resulting trees:
4 – The implementation

Resulting trees:
Contents

1. Why ‘surfacing’?
2. Our goal
3. The algorithm
4. Implementation
5. Discussion
5 – Discussion

1. Trigger
   a. Dependency (Alpino) \(\rightarrow\) *discontinuous* constituents
   b. Linguistic research desires surface-*continuous* constituents
      • Scrambling
      • Extraposition
      • Information structure
      • First-constituent research
      • Verb-cluster order
5 – Discussion

1. Trigger
   a. Dependency (Alpino) \( \rightarrow \) discontinuous constituents
   b. Linguistic research desires continuous constituents
      • Scrambling
      • Extraposition
      • Information structure
      • First-constituent research
      • Verb-cluster order

2. Solution: ‘surfacing’ algorithm
   a. Transform discontinuous to continuous
   b. Retain dependency constituent information
5 – Discussion

1. Trigger
   a. Dependency (Alpino) → *discontinuous* constituents
   b. Linguistic research desires *continuous* constituents
      • Scrambling
      • Extraposition
      • Information structure
      • First-constituent research
      • Verb-cluster order

2. Solution: ‘surfacing’ algorithm
   a. Transform discontinuous to continuous
   b. Retain dependency constituent information

3. Implementation
   a. Windows: ‘Cesax’
   b. C# (Win/Mac/Linux): ‘FoliaParse’
5 – Discussion

1. Trigger
   a. Dependency (Alpino) → discontinuous constituents
   b. Linguistic research desires continuous constituents
      • Scrambling
      • Extraposition
      • Information structure
      • First-constituent research
      • Verb-cluster order

2. Solution: ‘surfacing’ algorithm
   a. Transform discontinuous to continuous
   b. Retain dependency constituent information

3. Implementation
   a. Windows: ‘Cesax’
   b. C# (Win/Mac/Linux): ‘FoliaParse’

4. Result:
   a. Easier constituent-order phenomena research using Xpath/Xquery