Incremental Discontinuous Phrase Structure Parsing with the GAP Transition

Maximin Coavoux\textsuperscript{1,2} – Benoît Crabbé\textsuperscript{1,2,3}

\textsuperscript{1}Univ Paris Diderot – Sorbonne Paris Cité (SPC)  
\textsuperscript{2}Laboratoire de Linguistique Formelle (LLF, CNRS)  
\textsuperscript{3}Institut Universitaire de France (IUF)

EACL – Valencia – April 2017
Discontinuous Constituency Trees: Example

- Task: predict (potentially) discontinuous constituency trees

Translation: But nothing interesting happened (Negra Corpus)
Discontinuous Constituency Trees: What for?

Wide range of phenomena
- Long distance extractions: some relative clauses, questions
- Dislocations
- Cross serial dependencies

Syntactical discontinuities are rather frequent
- languages with some degree of word order flexibility: 30% of sentences in German treebanks (Maier and Lichte, 2009)
- configurational languages: 20% of sentences in Discontinuous Penn Treebank (Evang and Kallmeyer, 2011)

Annotation strategies:
- Use empty categories (traces), coindexation (PennTB)
- Use crossing branches (Negra, Tiger) → Discontinuous trees
Discontinuous Parsing

Standard constituency parsing focused on projective trees
- remove traces/empty categories: too hard
- projectivize trees

Approaches to discontinuous parsing
- Probabilistic grammar, CKY-like decoding
  - exact parsing has high polynomial complexity \( O(n^3 f) \)
  - does not scale to full corpora
  - limited accuracy
- Transition based methods
  - Easy first (Versley, 2014), Swap action (Maier, 2015)
  - faster, scalable
- Reduction to dependency parsing
  - tree conversion from const to dep
  - most successful approach so far
Discontinuous Parsing

Standard constituency parsing focused on projective trees
  ▶ remove traces/empty categories: too hard
  ▶ projectivize trees

Approaches to discontinuous parsing
  ▶ Probabilistic grammar, CKY-like decoding
    ▶ exact parsing has high polynomial complexity $\mathcal{O}(n^{3f})$
    ▶ does not scale to full corpora
    ▶ limited accuracy
Discontinuous Parsing

Standard constituency parsing focused on projective trees
- remove traces/empty categories: too hard
- projectivize trees

Approaches to discontinuous parsing
- Probabilistic grammar, CKY-like decoding
  - exact parsing has high polynomial complexity $O(n^{3f})$
  - does not scale to full corpora
  - limited accuracy
- Transition based methods
  - Easy first (Versley, 2014), Swap action (Maier, 2015)
  - faster, scalable
Discontinuous Parsing

Standard constituency parsing focused on projective trees
▶ remove traces/empty categories: too hard
▶ projectivize trees

Approaches to discontinuous parsing
▶ Probabilistic grammar, CKY-like decoding
  ▶ exact parsing has high polynomial complexity $\mathcal{O}(n^{3f})$
  ▶ does not scale to full corpora
  ▶ limited accuracy
▶ Transition based methods
  ▶ Easy first (Versley, 2014), Swap action (Maier, 2015)
  ▶ faster, scalable
▶ Reduction to dependency parsing
  ▶ tree conversion from const to dep
  ▶ most successful approach so far
Contributions

- New transition system for discontinuous constituency parsing
  **Shift-Reduce+Gap**
  - Approximate parsing (drop grammaticality constraints)
  - Efficient (linear time parsing)
  - State-of-the-art results on 2 German treebanks with a perceptron

- Empirical comparison with previous best transition system
  (Shift-Reduce+Swap; Maier, 2015)
Outline

Introduction

Transition based parsing

The GAP transition

Experiments

Discussion: Gap vs Swap
Transition-based Parsing: Standard Shift-Reduce

- Syntactic tree equivalent to a sequence of actions
- Classifier to predict actions
- Configuration = (Stack, Buffer)
  - Stack contains tree nodes
  - Buffer contains tokens
- Use actions to derive new configurations until the stack contains a single tree and the buffer is empty

Start configuration

<table>
<thead>
<tr>
<th>Stack</th>
<th>Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cats</td>
</tr>
<tr>
<td>NNS</td>
<td>MD</td>
</tr>
<tr>
<td>VB</td>
<td>can</td>
</tr>
<tr>
<td></td>
<td>meow</td>
</tr>
</tbody>
</table>

Terminal configuration

```
S
 /  \
NP  VP
 /   /
NNS MD VB
 /   /
Cats can meow
```

Actions

```
Stack

Buffer
```
Transition-based Parsing: Standard Shift-Reduce

- $s_i$, $b_i$: index elements in stack and buffer
- Reduce-Left-X, Reduce-Right-X for each non-terminal $X$
- Left/right: assign the head of the new constituent
  - useful because features use heads of constituents
Plan

Introduction

Transition based parsing

The **GAP** transition

Experiments

Discussion: Gap vs Swap
Extending the Shift-Reduce algorithm

Standard shift-reduce: reductions apply to the 2 topmost elements in the stack:

\[ \text{Reduce-X} \]

\[ \cdots \quad s_3 \quad s_2 \quad s_1 \quad s_0 \quad \cdots \]

\[ \uparrow \quad \uparrow \quad \ast \quad \ast \]

\[ \cdots \quad s_3 \quad s_2 \quad X \]

\[ s_1 \quad s_0 \]

\[ \rightarrow \text{can only derive projective trees} \]
Extending the Shift-Reduce algorithm

Standard shift-reduce: reductions apply to the 2 topmost elements in the stack:

\[
\vdots \quad s_3 \quad s_2 \quad s_1 \quad s_0 \\
\uparrow \quad \uparrow \\
* \quad * \\
\vdots \quad s_3 \quad s_2 \quad X
\]

→ can only derive projective trees
To handle **discontinuities**, allow reductions with \( s_0 \) and any other symbol in the stack. Side effect: **implicitly reordering terminals**

\[
\vdots \quad s_3 \quad s_2 \quad s_1 \quad s_0 \\
\uparrow \quad \uparrow \\
* \quad * \\
\vdots \quad s_3 \quad s_1 \quad X \\
\quad s_2 \quad s_0
\]
Shift-Reduce+Gap

- **GAP** action: access next non-terminal in the stack for a potential reduction
  - Choose dynamically which element in the stack is used for a reduction

```
... s4 s3 s2 s1 s0
```

```
? / \
*  *
```
GAP action: access next non-terminal in the stack for a potential reduction

- Choose dynamically which element in the stack is used for a reduction
GAP action: access next non-terminal in the stack for a potential reduction

Choose dynamically which element in the stack is used for a reduction
Shift-Reduce+Gap

- GAP action: access next non-terminal in the stack for a potential reduction
  - Choose dynamically which element in the stack is used for a reduction

```
\begin{array}{l}
\ldots \quad s_4 \quad s_3 \quad s_2 \quad s_1 \quad s_0 \\
\uparrow \\
* \\
\uparrow \\
* \\
\end{array}
```
The usual stack is split into 2 parts:

- A Stack $S$ (bottom part)
- A Deque $D$ (upper part)
- Reductions are always applied to $s_0$ and $d_0$
- (The buffer is still a buffer)
The **Gap** transition: **Stack Deque Buffer**

Projective case: $s_0$ and $d_0$ are the 2 topmost elements
The $\text{GAP}$ transition: Stack Deque Buffer

After 1 $\text{GAP}$
The GAP transition: Stack Deque Buffer

After 2 GAPs (Can gap as long as length of $S$ is $> 1$)
The $\text{GAP}$ transition: Stack Deque Buffer

Reduction: pick $s_0$ and $d_0$
The **GAP** transition: Stack Deque Buffer

Reduction to $X$: create new node

```
<table>
<thead>
<tr>
<th>x6</th>
<th>x5</th>
<th>x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>x2</td>
<td>x1</td>
<td></td>
</tr>
</tbody>
</table>

$X_3$  $X_0$

...
The **GAP** transition: **Stack Deque Buffer**

Reduction to $X$: flush $D$ to $S$
The GAP transition: Stack Deque Buffer

Reduction to $X$: push new node to $D$
Now, top of $S$ is $x_1$ and top of $D$ is $X$
Transition-based Parsing: Shift-Reduce+Gap

Configuration = (Stack, Deque, Buffer)
- Initial configuration = (∅, ∅, [w₁, w₂ ... wₙ])
- Final configuration = (∅, [A], ∅)
  - A = axiom

Transition set

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift (S, D, b₀</td>
<td>(S</td>
</tr>
<tr>
<td>Reduce-Left/Right(X) (S</td>
<td>s₀, D</td>
</tr>
<tr>
<td>Gap (S</td>
<td>s₀, D, B)</td>
</tr>
</tbody>
</table>

Let’s see a full example . . .
Shift-Reduce-Gap: Stack – Deque – Buffer

Initialisation [phrase from DPTB, Evang and Kallmeyer, 2011]
Shift-Reduce-Gap: Stack – Deque – Buffer

Shift

NP

NP

PP

SBAR@S

VP

NP

WHADVP

IN

WRB

NNS

NNS

VBP

Sh

JJ*

lucid

explanations

of

how

computers

work
Shift-Reduce-Gap: Stack – Deque – Buffer

Shift

NP

PP

SBAR@S

VP

NP

WHADVP

IN

WRB

NNS

NNS

VBP

Sh, Sh

lucid explanations of how computers work
Shift-Reduce-Gap: Stack – Deque – Buffer

Reduce-NP

```
NP
| PP
| SBAR@S
| VP
| WHADVP
| NP
| IN
| WRB
| NNS
| VBP
| work
```

```
NP*
| JJ
| NNS
| lucid
| explanations

Sh, Sh, R(NP)
```
Shift-Reduce-Gap: **Stack – Deque – Buffer**

**Shift**

- **NP**
  - **NP**
    - **IN***
      - **JJ** *lucid*
      - **NNS** *explanations*
    - **of**
  - **WHADVP**
    - **WRB** *how*
    - **NNS** *computers*
  - **VP**
    - **SBAR@S**
      - **PP**
        - **VBP** *work*

Sh, Sh, R(NP), Sh
Shift-Reduce-Gap: Stack – Deque – Buffer

Shift

**NP**

**IN***

**NP**

**WRB***

**NP**

**VBP**

Sh, Sh, R(NP), Sh, Sh

lucid explanations of how computers work
Shift-Reduce-Gap: Stack – Deque – Buffer

ReduceUnary-WHADVP

NP

<table>
<thead>
<tr>
<th>JJ</th>
<th>NNS</th>
<th>IN*</th>
<th>WHADVP*</th>
<th>NNS</th>
<th>VBP</th>
</tr>
</thead>
<tbody>
<tr>
<td>lucid</td>
<td>explanations</td>
<td>of</td>
<td>how</td>
<td>computers</td>
<td>work</td>
</tr>
</tbody>
</table>

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP)
Shift-Reduce-Gap: Stack – Deque – Buffer

Shift

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP), Sh
Shift-Reduce-Gap: Stack – Deque – Buffer

ReduceUnary-NP

NP

PP

SBAR@S

VP

WHADVP*

NP*

VP

IN

WRB

NNS

VBP

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP), Sh, RU(NP)
Shift-Reduce-Gap: **Stack – Deque – Buffer**

Shift

```
NP
  └── PP
    └── SBAR@S
      └── VP
          └── VBP*
                └── work

NP
  └── WHADVP
    └── NP*

IN
  └── WRB
    └── NNS
        └── computers

JJ
  └── lucid

NNS
  └── explanations

IN
  └── of

NNS
  └── how
```

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP), Sh, RU(NP), Sh
Shift-Reduce-Gap: Stack – Deque – Buffer

Gap

NP

PP

SBAR@S

VP

WHADVP*

NP

VBP*

JJ  NNS  IN  WRB  NNS  VBP*

lucid  explanations  of  how  computers  work

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP), Sh, RU(NP), Sh, Gap
Shift-Reduce-Gap: Stack – Deque – Buffer

Reduce-VP

```
NP
  NP
    JJ  NNS
    lucid  explanations
  IN
  of

WHADVP
  WRB
  how

NP*
  NNS
  computers

VP*
  VBP
  work
```

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP), Sh, RU(NP), Sh, Gap, R(VP)
Shift-Reduce-Gap: Stack – Deque – Buffer

NP

IN*

NP

JJ

lucid

NNS

explanations

IN*
of

WHADVP

WRB

how

NNS

computers

VP

SBAR@S*

PP

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP), Sh, RU(NP), Sh, Gap, R(VP), R(S)
Shift-Reduce-Gap: Stack – Deque – Buffer

Reduce-PP

NP

PP*

SBAR@S

VP

WHADVP

NP

VBP

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP), Sh, RU(NP), Sh, Gap, R(VP), R(S), R(PP)
Shift-Reduce-Gap: Stack – Deque – Buffer

Reduce-NP

NP*

PP

SBAR@S

VP

WHADVP

NP

VBP

Sh, Sh, R(NP), Sh, Sh, RU(WHADVP), Sh, RU(NP), Sh, Gap, R(VP), R(S), R(PP), R(NP)
Some properties of Shift-Reduce+Gap

- Derives any labelled discontinuous tree over a set of non-terminal symbols
  - handles well-nested and ill-nested trees
- With some (easily checked) constraints on actions: always outputs a tree
- Longest derivation for a sentence of size $n$ is in $O(n^2)$
  - In practice, parsing in linear time: limited number of discontinuities in datasets
- Related to Covington’s (2001) non-projective dependency parsing algorithm
  - transition system with 3 data structures (Gómez-Rodríguez and Fernández-González, 2015)
Plan

Introduction

Transition based parsing

The GAP transition

Experiments

Discussion: Gap vs Swap
Experiments: Data

- 2 German corpora:
  - Tiger Corpus (Brants et al., 2002), \( \approx 50000 \) sentences
  - Negra Corpus (Skut et al., 1997), \( \approx 20000 \) sentences

- Both were natively annotated with discontinuous constituents

- \( \approx 30\% \) of sentences contain at least one discontinuity

- Preprocessing
  - Head annotation with headrules
  - Head-outward binarization (+ order-0 Markovization)
  - Reattach punctuation locally (avoid spurious discontinuity)

- Assume tags are available (either gold or predicted)
Experiments: Classifier

- Deterministic oracle to transform gold trees to gold sequences of actions
- Simple averaged structured perceptron
  - Beam search
  - Early update
- Perceptron is biased towards longer derivations
  - padding derivations with IDLE actions (Zhu et al., 2013) to improve comparability between derivations in the beam
- C++ implementation: github.com/mcoavoue/mtg
  - Scalable to full corpora, 4700 tokens/s with beam size = 4
  - Tree structured stack (TSS) for compact representation of beam
Experiments: Feature templates

3 sets of feature templates

- **Baseline**, 40 templates: standard features for a projective constituency parser (Zhu et al., 2013)

- **+Extended**, 52 templates: adds information about
  - gapped elements \((d_1, d_2)\)
  - extended context \((s_3)\)

- **+Spans**, 87 templates: adds information about constituent boundaries (Hall et al., 2014)
  - e.g. leftmost/rightmost terminal spanned by \(s_0\), etc..
Experiments: Results (Gold POS) – Internal comparisons

Disc. F1 : discontinuous constituents only. Evaluator: discodop (Van Cranenburgh et al., 2016)

<table>
<thead>
<tr>
<th>Beam size Gap, +Spans</th>
<th>TigerHN8 (dev) F1</th>
<th>Disc. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>81.86</td>
<td>48.49</td>
</tr>
<tr>
<td>4</td>
<td>83.27</td>
<td>53.00</td>
</tr>
<tr>
<td>8</td>
<td>83.61</td>
<td>54.42</td>
</tr>
<tr>
<td>16</td>
<td>83.84</td>
<td>54.81</td>
</tr>
<tr>
<td><strong>32</strong></td>
<td><strong>84.32</strong></td>
<td><strong>56.22</strong></td>
</tr>
<tr>
<td>64</td>
<td>84.14</td>
<td>56.01</td>
</tr>
<tr>
<td>128</td>
<td>84.05</td>
<td>55.76</td>
</tr>
</tbody>
</table>

- Beam size helpful for disc. constituents. From 2 to 32:
  - + 8 for discontinuous constituents
  - + 3 for all constituents
  - Search compensates for lack of global view
### Experiments: Results (Gold POS) – Internal comparisons

<table>
<thead>
<tr>
<th>Test</th>
<th>TigerHN08</th>
<th>Improvement over baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>Disc. F1</td>
</tr>
<tr>
<td>Beam = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift-Reduce+Gap, <strong>Baseline</strong></td>
<td>81.67</td>
<td>44.83</td>
</tr>
<tr>
<td>Shift-Reduce+Gap, <strong>Extended</strong></td>
<td>82.43</td>
<td>48.81</td>
</tr>
<tr>
<td>Shift-Reduce+Gap, <strong>Spans</strong></td>
<td><strong>83.16</strong></td>
<td><strong>49.76</strong></td>
</tr>
</tbody>
</table>

- **+Extended**: information about content of gap is useful especially for discontinuities
- **+Spans**: information about constituent boundaries is useful
Experiments: Results (Gold POS) – External comparisons

Comparison with chart parsers

- CKY-like decoding
- Kallmeyer and Maier (2013): Probabilistic LCFRS
- Van Cranenburgh et al. (2016): DOP model

<table>
<thead>
<tr>
<th>Negra sentence length</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kallmeyer and Maier, 2013</td>
<td>≤ 30</td>
</tr>
<tr>
<td>Van Cranenburgh et al., 2016</td>
<td>≤ 40</td>
</tr>
<tr>
<td>Shift-Reduce+Gap, beam=32</td>
<td>All</td>
</tr>
</tbody>
</table>
Experiments: Results (Gold POS) – External comparisons

Comparison with transition based parsers

- Maier 2015: extends shift-reduce with swap action
  - Swap: push 2nd element of stack back onto the buffer (adapted from d-parsing)
  - Same setting as ours: global perceptron with beam search

<table>
<thead>
<tr>
<th></th>
<th>beam size</th>
<th>F1</th>
<th>Disc.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maier 2015</td>
<td>4</td>
<td>74.71</td>
<td>18.8</td>
<td></td>
</tr>
<tr>
<td>Maier and Lichte 2016</td>
<td>4</td>
<td>76.46</td>
<td>16.3</td>
<td></td>
</tr>
<tr>
<td>Shift-Reduce+Gap</td>
<td>4</td>
<td>80.40</td>
<td>46.5</td>
<td></td>
</tr>
<tr>
<td>Shift-Reduce+Gap</td>
<td>32</td>
<td><strong>81.60</strong></td>
<td><strong>49.2</strong></td>
<td></td>
</tr>
</tbody>
</table>
Experiments: Results (Gold POS) – External comparisons

Comparison with dependency parsing based methods

Fernández-González and Martins (2015) *Parsing as Reduction*

1. Convert trees to (non-projective) dependency trees
2. Parse with dependency parser
3. Convert result to discontinuous constituency trees

<table>
<thead>
<tr>
<th></th>
<th>Negra</th>
<th>TigerHN8</th>
<th>TigerSPMRL</th>
<th>Predicted POS TigerSPMRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeMa2015</td>
<td>80.5</td>
<td><strong>84.2</strong></td>
<td>80.6</td>
<td>77.3</td>
</tr>
<tr>
<td>SR+Gap</td>
<td><strong>82.2</strong></td>
<td>84.0</td>
<td><strong>81.5</strong></td>
<td><strong>79.3</strong></td>
</tr>
</tbody>
</table>
Plan

Introduction

Transition based parsing

The \textsc{Gap} transition

Experiments

Discussion: Gap vs Swap
Discussion: Gap vs Swap

Swap and Gap

- both extensions of shift-reduce
- both $O(n^2)$
- but rather large difference in accuracies (in comparable experimental settings).

Why?

- Main hypothesis: Shift-Reduce+Gap tends to produce much shorter derivations (easier to learn)
Discussion: Gap vs Swap

**Swap action**: push the second element of the stack back onto the buffer. (It must be a terminal)

Swap transition system (Maier, 2015) uses 2 usual data structures (stack + buffer).
We expect the Gap transition system to produce shorter derivation than Swap in general.

1. Each swapped terminal must be shifted again (and might be reswapped and reshifted if need be)

2. Both systems are implicitly predicting a reordering of terminals that would make a tree projective
   - Swap *terminals* only $\rightarrow$ reordering terminals not efficient
   - Can gap *whole subtrees* $\rightarrow$ reordering more efficient
### Discussion: Derivation Length

Some measures on the Tiger corpus (train) using an oracle:

<table>
<thead>
<tr>
<th></th>
<th>SR+Gap</th>
<th>SR+Swap</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average derivation length wrt</strong> $n$</td>
<td>$2.03n$</td>
<td>$3.09n$</td>
</tr>
<tr>
<td>Longest derivation</td>
<td>276</td>
<td>2187</td>
</tr>
<tr>
<td>Total number of gaps/swaps</td>
<td>64096</td>
<td>411970</td>
</tr>
<tr>
<td>Max consecutive gaps/swaps</td>
<td>10</td>
<td>69</td>
</tr>
</tbody>
</table>
Conclusion

Summary

- New transition system for efficient and accurate discontinuous parsing
- State-of-the-art results on German treebanks

Perspectives

- Application to other languages (English, French) and other types of linguistic discontinuities (MWE?)
- bi-LSTM encoder (Cross and Huang, 2016)
- Dynamic oracle?
- Relationship to LCFRS automata?
Thanks!

Comments? Questions?

Thanks to Chloé Braud, Héctor Martínez Alonso, Djamé Seddah, Olga Seminck
Transition System

**Input**

\[ t_1[w_1]t_2[w_2] \ldots t_n[w_n] \]

**Axiom**

\[ \langle \epsilon, \epsilon, t_1[w_1]t_2[w_2] \ldots t_n[w_n] \rangle \]

**Goal**

\[ \langle \epsilon, S[w], \epsilon \rangle \]

**SHIFT**

\[
\begin{align*}
\frac{\langle S, D, t[w] \mid B \rangle}{\langle S \mid D, t[w], B \rangle} & \\
\frac{\langle S \mid D, t[w], B \rangle}{\langle S \mid D, t[w] \mid B \rangle}
\end{align*}
\]

**REDUCE-UNARY(X)**

\[
\begin{align*}
\frac{\langle S, d_0[h], B \rangle}{\langle S, X[h], B \rangle} & \\
\frac{\langle S, d_0[h], B \rangle}{\langle S, X[h], B \rangle}
\end{align*}
\]

**REDUCE-RIGHT(X)**

\[
\begin{align*}
\frac{\langle S \mid s_0[h], D \mid d_0[h'], B \rangle}{\langle S \mid D, X[h'], B \rangle} & \\
\frac{\langle S \mid D, X[h'], B \rangle}{\langle S \mid D, X[h'], B \rangle}
\end{align*}
\]

**REDUCE-LEFT(X)**

\[
\begin{align*}
\frac{\langle S \mid s_0[h], D \mid d_0[h'], B \rangle}{\langle S \mid D, X[h'], B \rangle} & \\
\frac{\langle S \mid D, X[h'], B \rangle}{\langle S \mid D, X[h'], B \rangle}
\end{align*}
\]

**GAP**

\[
\begin{align*}
\frac{\langle S \mid s_0[h], D, B \rangle}{\langle S, s_0[h] \mid D, B \rangle} & \\
\frac{\langle S, s_0[h] \mid D, B \rangle}{\langle S, s_0[h] \mid D, B \rangle}
\end{align*}
\]
### Experiments: Results with predicted pos

<table>
<thead>
<tr>
<th>Method</th>
<th>F1 (spmrl.prm)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>≤ 70</strong></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td></td>
</tr>
<tr>
<td>Versley (2014), EasyFirst</td>
<td>73.90</td>
</tr>
<tr>
<td>Fernández-González and Martins (2015)</td>
<td>77.72</td>
</tr>
<tr>
<td>SR-GAP, beam=32, +Spans</td>
<td>79.44</td>
</tr>
<tr>
<td>SR-GAP, greedy, bi-lstms, own tagging</td>
<td><strong>82.22</strong></td>
</tr>
</tbody>
</table>

- + 3 over structured perceptron
Shift-Reduce+Gap is related to Covington’s (2001) (family of) algorithm(s) for unrestricted dependency parsing

**Algorithm 1** Covington’s algorithm for d-parsing $O(n^2)$

1: for $i = 1$ to $n$ do
2:   for $j = i - 1$ downto 1 do
3:     link(i,j)
4:   end for
5: end for

`Link(i,j)`: either add arc $i \rightarrow j$, or arc $j \rightarrow i$ or do nothing.

- Can be formulated as a transition system with 3 data structures
Es bestünde somit hinreichender Spielraum

* indicates head percolation