Stockholm

TreeAligner

Building and Searching Parallel Treebanks with the Stockholm TreeAligner

Torsten Marek

<torsten.marek@gmx.net>

Universität des Saarlandes

June 28th, 2008

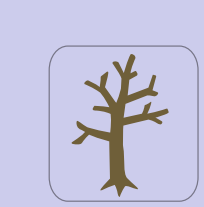
Outline

1. Introduction to Parallel Treebanks
2. (A quite incomplete introduction to) the TIGER Query Language
3. Query Language Extensions
4. Overview of the Query Evaluation Engine
5. Future Plans & Projects

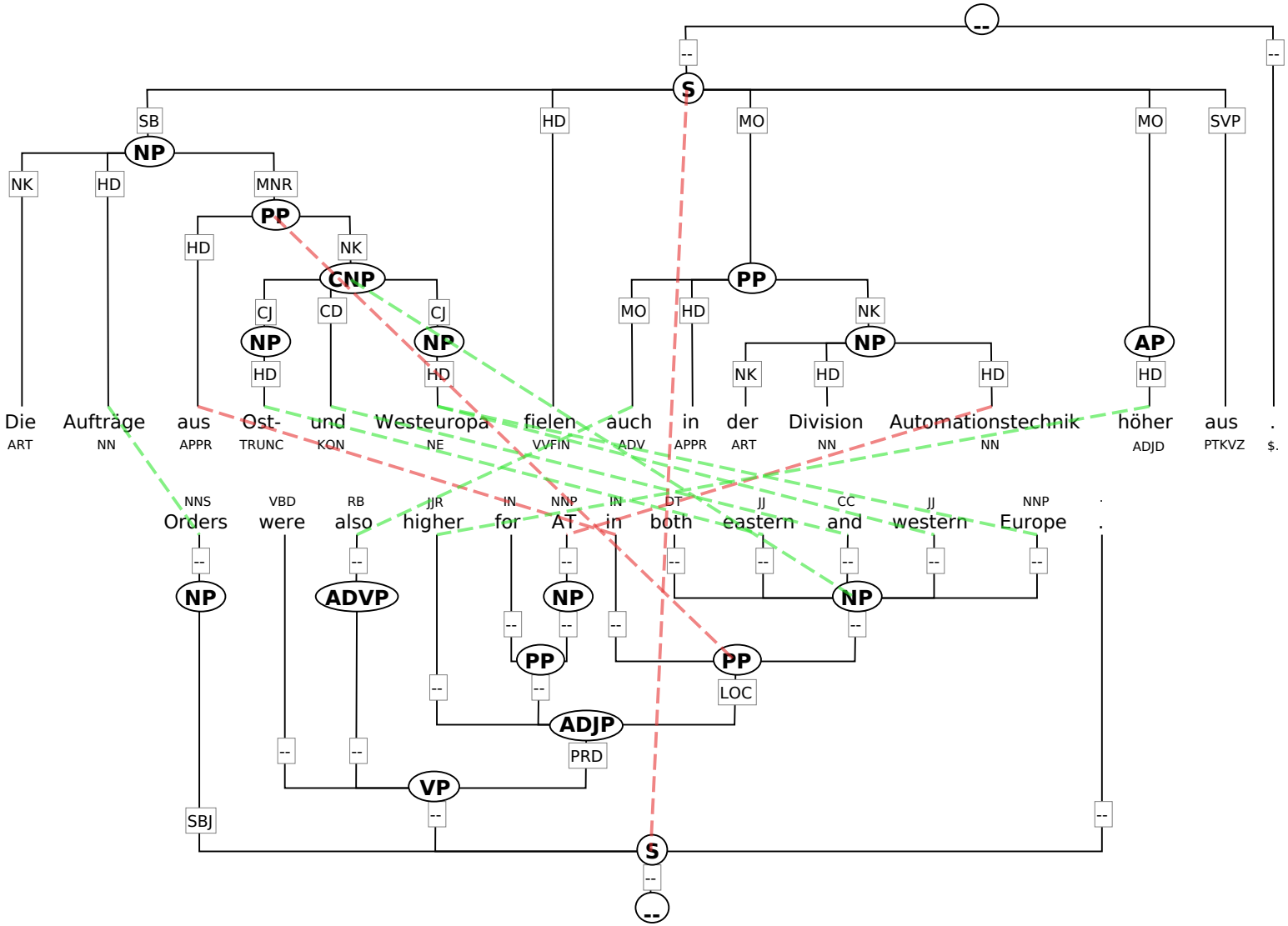
Parallel Treebanks

Definition

- the same text
- in several languages
- with syntactical annotations
- and varying levels of alignments
  - sentences
  - words
  - (linguistic) phrases



Phrase Alignment Example



Creating Parallel Treebanks

Alignment Guidelines

1. Align items that can be re-used as units in a machine translation system
2. Align as many items (i.e. words and phrases) as possible
3. Align as close as possible to the tokens

(Samuelsson and Volk, 2006) + upcoming HJCL paper

Creating Parallel Treebanks

Alignment Guidelines

1. Align items that can be re-used as units in a machine translation system
2. Align as many items (i.e. words and phrases) as possible
3. Align as close as possible to the tokens

(Samuelsson and Volk, 2006) + upcoming HJCL paper

Fuzzy Alignments

"mehr als eine Maschine" vs. *"more than a piece of hardware"*
not literal translation, but meaning is intended to be the same.

SMULTRON

Creators

- Created by Martin Volk's group at Stockholm University, started in 2005
- Custom tool for annotating syntactical alignments: Stockholm TreeAligner (Lundborg et al., 2007)
- Available from `http://www.ling.su.se/dali/research/smultron/index.htm`
- free of charge for research purposes

SMULTRON

Creators

- Created by Martin Volk's group at Stockholm University, started in 2005
- Custom tool for annotating syntactical alignments: Stockholm TreeAligner (Lundborg et al., 2007)
- Available from `http://www.ling.su.se/dali/research/smultron/index.htm`
- free of charge for research purposes

Proportions

- 1000 sentences German, English, Swedish
  - 500 sentences from "Sophie's World"
  - 500 sentences from business + advertisement texts
- all language pairs aligned

The Shortest Possible TIGER Primer

The Shortest Possible TIGER Primer

TIGER

Syntactically annotated treebank of German newspaper text, 50,000+ sentences.

The Shortest Possible TIGER Primer

TIGER

Syntactically annotated treebank of German newspaper text, 50,000+ sentences.

TIGER-XML

XML format used to store the TIGER corpus

The Shortest Possible TIGER Primer

TIGER

Syntactically annotated treebank of German newspaper text, 50,000+ sentences.

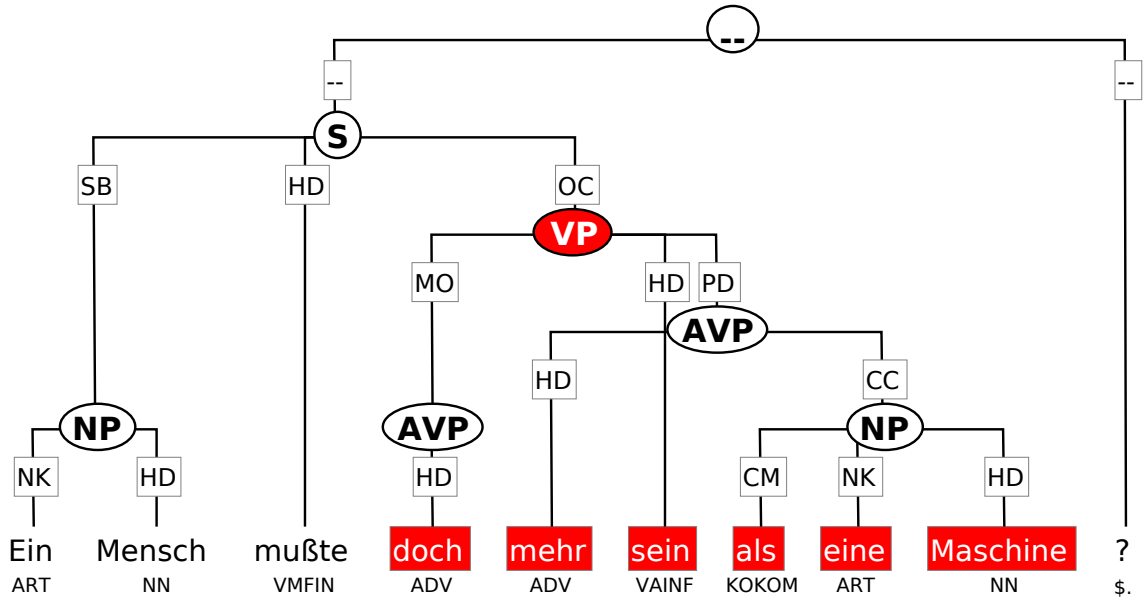
TIGER-XML

XML format used to store the TIGER corpus

TIGERSearch

A query language & visualization tool for treebanks (Lezius, 2002)

TIGER Graphs



Graph Type

- directed
- acyclic
- no structure sharing
- discontinuous fringes

TIGER Graphs

pgflastimage

Graph Type

- directed
- acyclic
- no structure sharing
- discontinuous fringes

TIGER Query Example

A Sample Query

Find all NPs that directly dominate the noun "Haus".

TIGER Query Example

A Sample Query

Find all NPs that directly dominate the noun "Haus".

. . . formalized

```
[cat="NP"] > [word="Haus" & pos="NN"]
```

TIGER Query Example

A Sample Query

Find all NPs that directly dominate the noun "Haus".

. . . formalized

```
[cat="NP"] > [word="Haus" & pos="NN"]
```

. . . and rendered

Node Descriptions

Semantics: `[cat="NP"]`

Boolean expressions of feature value constraints, to select terminal (word) or non-terminal (phrasal) nodes from syntax graphs.

Node Descriptions

Semantics: `[cat="NP"]`

Boolean expressions of feature value constraints, to select terminal (word) or non-terminal (phrasal) nodes from syntax graphs.

A Complex Node Description

Find feminine nouns that are dative or accusative plural:

```
[case="Acc"|"Dat" & gender="Fem" & number="Pl" & pos="NN"]
```

Node Descriptions

Semantics: `[cat="NP"]`

Boolean expressions of feature value constraints, to select terminal (word) or non-terminal (phrasal) nodes from syntax graphs.

A Complex Node Description

Find feminine nouns that are dative or accusative plural:

```
[case="Acc"|"Dat" & gender="Fem" & number="Pl" & pos="NN"]
```

Variables

All node descriptions can be assigned to a variable, can be reused in other parts of the query.

Node Predicates

Non-local node features

Predicates express non-local node features that are difficult to query for:

- *root*ness
- leaf discontinuity
- (token) arity

Node Predicates

Non-local node features

Predicates express non-local node features that are difficult to query for:

- *root*ness
- leaf discontinuity
- (token) arity

Example

Find all NPs that have exactly 3 terminal successors:

```
#np:[cat="NP"] & tokenarity(#np, 3)
```

Node Relations

Constrain the relation between two given nodes:

Node Relations

Constrain the relation between two given nodes:

- Dominance: `[cat="NP"] > [pos="ART"]`

Node Relations

Constrain the relation between two given nodes:

- Dominance: `[cat="NP"] > [pos="ART"]`
- Precedence: `[word="Peter"] . [word="Wolf"]`

Node Relations

Constrain the relation between two given nodes:

- Dominance: `[cat="NP"] > [pos="ART"]`
- Precedence: `[word="Peter"] . [word="Wolf"]`
- Siblings: `[word="sieben"] $ [word="Zwerge"]`

Node Relations

Constrain the relation between two given nodes:

- Dominance: `[cat="NP"] > [pos="ART"]`
- Precedence: `[word="Peter"] . [word="Wolf"]`
- Siblings: `[word="sieben"] $ [word="Zwerge"]`

Operator Modifiers

Behavior of relation operators can be modified (negation, dominance labels, transitivity).

The Alignment Query Language

The Alignment Query Language

Extending TIGER for Parallel Corpora

(Mettler, 2007) describes an extension of the TIGER query language to aligned parallel treebanks.

The Alignment Query Language

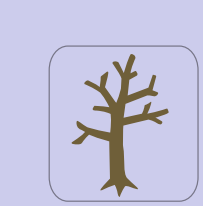
Extending TIGER for Parallel Corpora

(Mettler, 2007) describes an extension of the TIGER query language to aligned parallel treebanks.

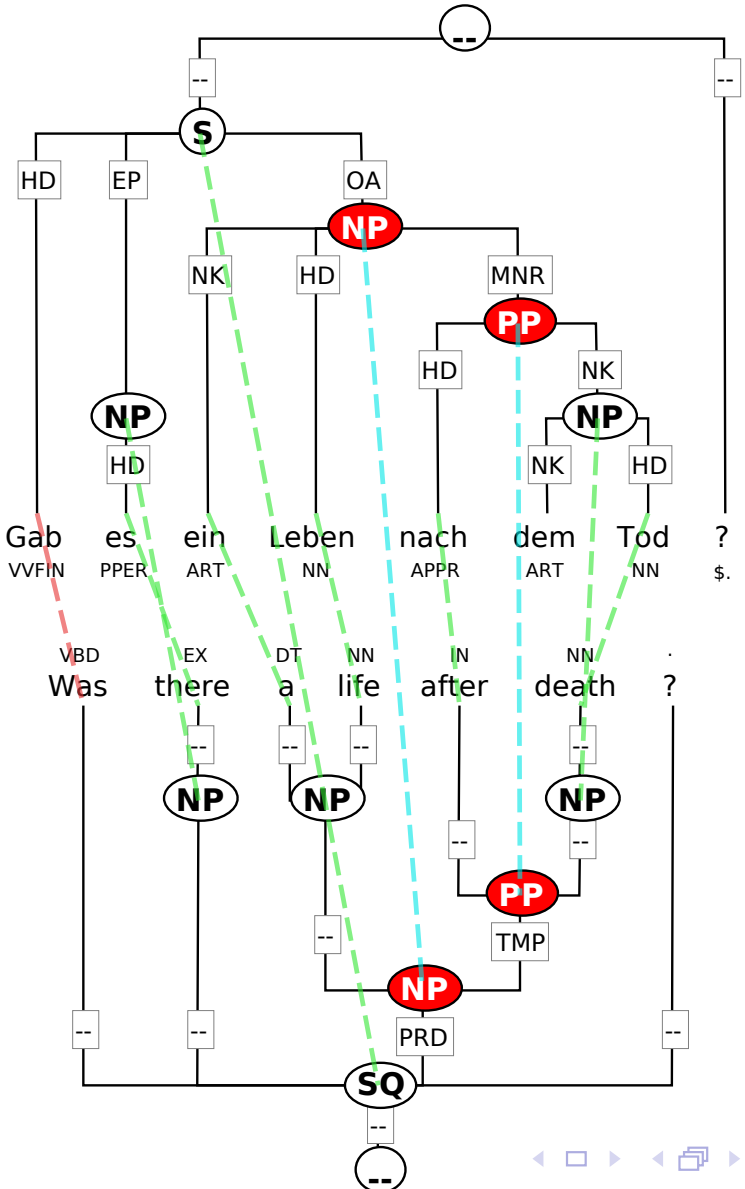
Sample Alignment Query

Find all PP-attachments to NPs that are aligned to each other:

```
de:  #np1:[cat="NP"] > #pp1:[cat="PP"]
en:  #np2:[cat="NP"] > #pp2:[cat="PP"]
#np1 * #np2 & #pp1 * #pp2
```



Sample AQL Result



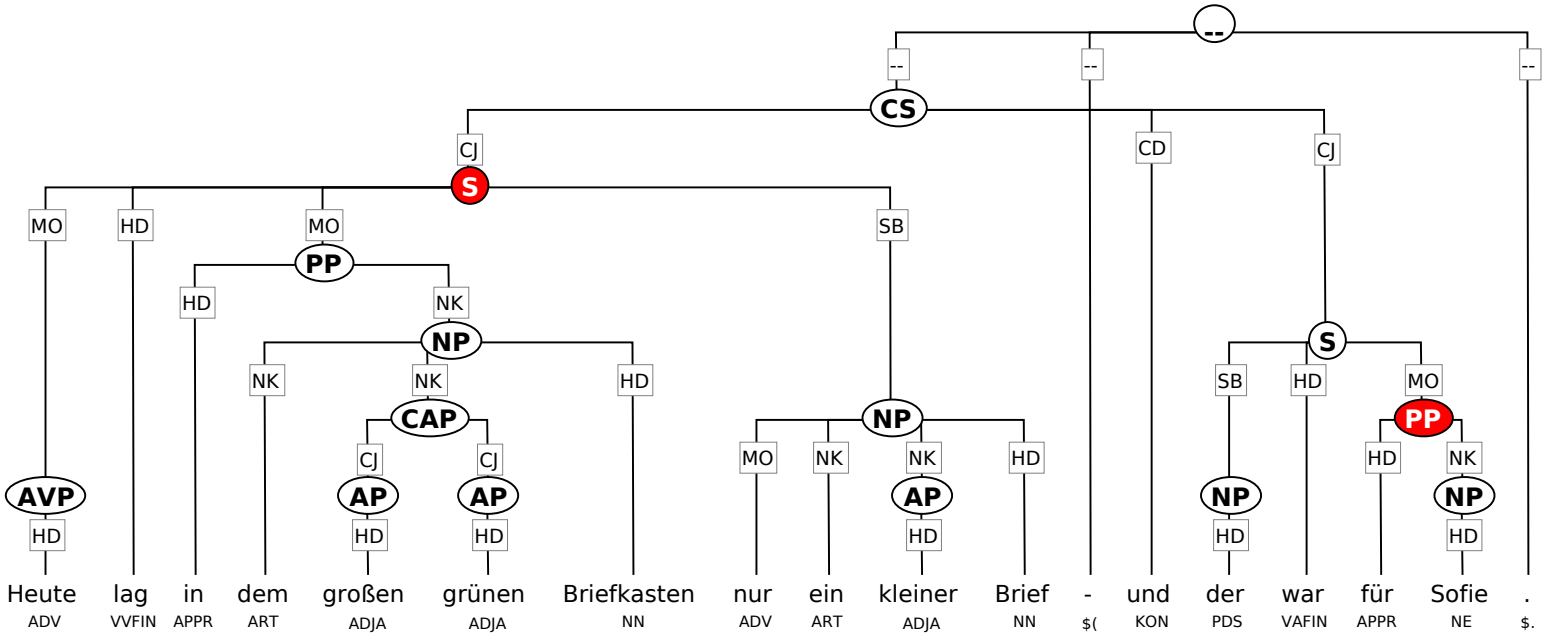
TIGER Limitations

Problematic Queries

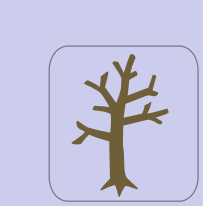
Find sentences that do not contain any PP:

```
[cat="S"] !>* [cat="PP"]
```

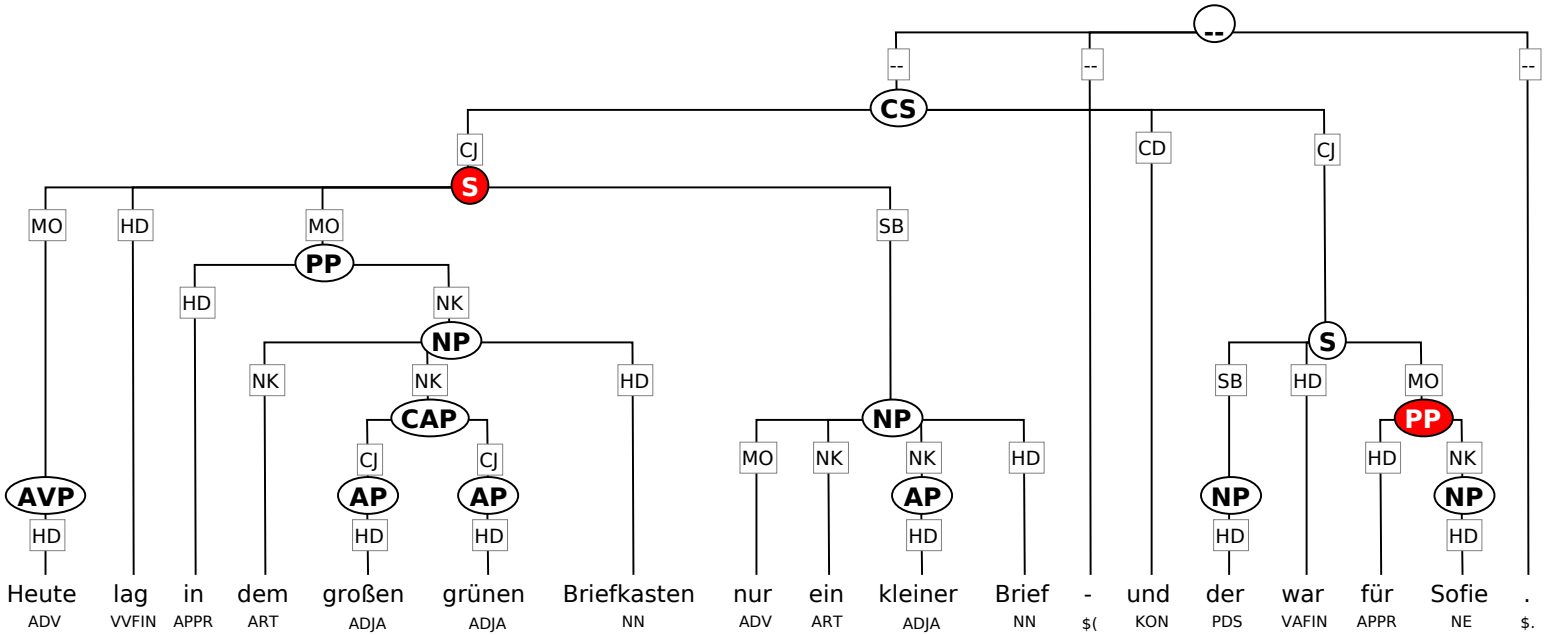
Wrong Results



- this result is not what we expected



Wrong Results



- this result is not what we expected
- but it is what we queried for:
  Find an S node and a PP node, such that the S node does not dominate the PP node

TIGER Limitations

Node Quantifiers

- all nodes are implicitly existentially qualified
  $\exists$`#np1` ($\exists$`#pp1` (`#np1` $>$ `#pp1` & ...))

TIGER Limitations

Node Quantifiers

- all nodes are implicitly existentially qualified
  $\exists$`#np1` ($\exists$`#pp1` (`#np1` $>$ `#pp1` & ...))
- no generalizations over multiple nodes in the same graph

Introducing Universal Quantification

Introducing Universal Quantification

Node Sets

- introduction of node *sets* rather than individual nodes
- constraints over node sets have to hold for all elements

Introducing Universal Quantification

Node Sets

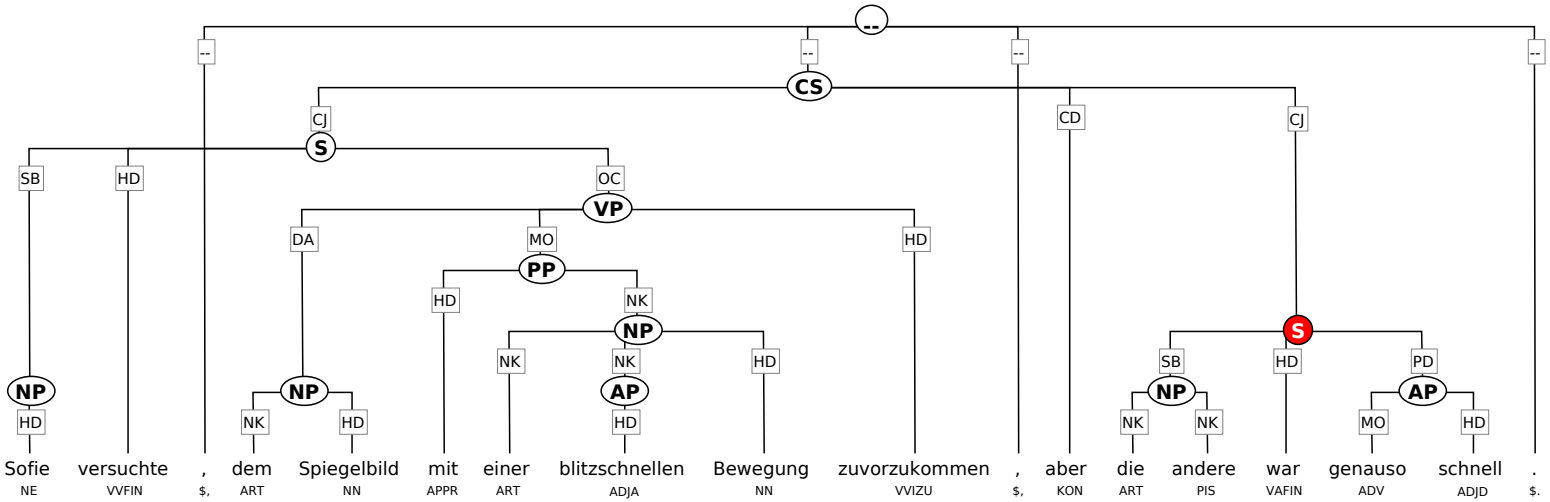
- introduction of node *sets* rather than individual nodes
- constraints over node sets have to hold for all elements

Second try

Find sentences that do not contain any PP:

```
[cat="S"] !>* %pp:[cat="PP"]
```

Correct Results



- implemented in the development version of STA
- described in (Marek et al., 2008) [to appear]

TIGER Query Evaluation in STA

Evaluation

Take a TIGER query and find all matching graphs:

1. Query Analysis
   - parse TIGER query
   - check for semantic errors
   - extract node descriptions, predicates and constraints
2. Node Candidate Retrieval
3. Relation Constraint Checking

Node Retrieval

Node Index

When a TIGER-XML corpus is first parsed, a complete index over all nodes is created using an SQL database:

- feature values
- computed node features (number of children etc)
- edge labels

Node Retrieval

Node Index

When a TIGER-XML corpus is first parsed, a complete index over all nodes is created using an SQL database:

- feature values
- computed node features (number of children etc)
- edge labels

Node Searching

1. convert feature value constraints to SQL statement

Node Retrieval

Node Index

When a TIGER-XML corpus is first parsed, a complete index over all nodes is created using an SQL database:

- feature values
- computed node features (number of children etc)
- edge labels

Node Searching

1. convert feature value constraints to SQL statement
2. execute query against the database

Node Retrieval

Node Index

When a TIGER-XML corpus is first parsed, a complete index over all nodes is created using an SQL database:

- feature values
- computed node features (number of children etc)
- edge labels

Node Searching

1. convert feature value constraints to SQL statement
2. execute query against the database
3. keep graphs for which each node description has at least one candidate

Relation Constraint Checking

Complexity

Each node description can have several matches in a graph. Given the results for the node descriptions $n_1 \ldots n_k$, there are

$$N = \prod_{i=1}^{k} |n_i|$$

possible combinations that have to be checked to see whether they satisfy the relation constraints.

Relation Checking Algorithms

Simplistic Algorithm

$\Rightarrow$ for all graphs: $O(n)$

$\Rightarrow$ for all combinations of nodes: $O(n^c)$

$\Rightarrow$ check all constraints on each combination: $O(n)$

Relation Checking Algorithms

Simplistic Algorithm

$\Rightarrow$ for all graphs: $O(n)$

$\Rightarrow$ for all combinations of nodes: $O(n^c)$

$\Rightarrow$ check all constraints on each combination: $O(n)$

Improvements

- only evaluate constraint once for each node pair
- fail as early as possible
  $\longrightarrow$ smart ordering of constraints
- try to prune candidate graphs as early as possible
- be lazy
  $\longrightarrow$ create *actual* results only if needed

Recap

- Parallel treebanks
- TIGER query language extensions
  - alignment queries for parallel treebanks
  - universal quantification
- TIGER query evaluation engine

References

Lezius, W. (2002).
Ein Suchwerkzeug für syntaktisch annotierte Korpora.
PhD thesis, IMS, University of Stuttgart.

Lundborg, J., Marek, T., Mettler, M., and Volk, M. (2007).
Using the Stockholm TreeAligner.
In *Proc. of The 6th Workshop on Treebanks and Linguistic Theories*, Bergen.

Marek, T., Lundborg, J., and Volk, M. (2008).
Extending the TIGER query language with universal quantification.
In *Proceedings of KONVENS '08 [to appear]*.

Mettler, M. (2007).
Parallel treebank search - The implementation of the Stockholm TreeAligner search.
C-uppsats, Stockholm University.

Samuelsson, Y. and Volk, M. (2006).
Phrase alignment in parallel treebanks.
In Hajic, J. and Nivre, J., editors, *Proc. of the Fifth Workshop on Treebanks and Linguistic Theories*, page 91–102, Prague.

TreeAligner Projects

Advertising

We are always looking for students who want to write their software project or thesis with us!

TreeDiff GUI

Background

The SMULTRON team needs a tool to visualize differences between different versions of their TIGER corpora.
A tree difference algorithm for TIGER syntax graphs has been developed in a thesis by a student from Stockholms University.

TreeDiff GUI

Background

The SMULTRON team needs a tool to visualize differences between different versions of their TIGER corpora.
A tree difference algorithm for TIGER syntax graphs has been developed in a thesis by a student from Stockholms University.

Idea

Implement a UI for displaying two versions of the same treebanks side-by-side or on top of each other and use the output of the TreeDiff algorithm to highlight the differences between the trees.

Extensions to the Query Engine

Implement Missing TIGER Features

- Feature value variables
- Toplevel disjunctions
- Better semantic analysis of queries, reject invalid ones

Extensions to the Query Engine

Implement Missing TIGER Features

- Feature value variables
- Toplevel disjunctions
- Better semantic analysis of queries, reject invalid ones

Beyond TIGER

- more universal quantification
- extended Alignment query language

TIGERSearch-like Browser

Background

TIGERSearch is showing its age (installer restricted to work with Java 1.4, broken on recent Linux systems, not maintained any further).

TIGERSearch-like Browser

Background

TIGERSearch is showing its age (installer restricted to work with Java 1.4, broken on recent Linux systems, not maintained any further).

Idea

Create a program that allows to browse and query monolingual treebanks similar to TIGERSearch.

Browser-based Corpus Queries

Idea

Create a web application that allows navigating and querying mono- and multilingual TIGER corpora from within a web browser. The installation of STA is cumbersome on Windows, because several libraries are needed.

Browser-based Corpus Queries

Idea

Create a web application that allows navigating and querying mono- and multilingual TIGER corpora from within a web browser. The installation of STA is cumbersome on Windows, because several libraries are needed.

Tasks

1. headless rendering to SVG/PNG (done)
2. query server
3. AJAX browser application

Thank You for Your Attention

Try it out!

`http://dev.ling.su.se/treealigner`

Questions?