Plug Latent Structures and Play Coreference Resolution

Sebastian Martschat, **Patrick Claus** and **Michael Strube**
Heidelberg Institute for Theoretical Studies gGmbH

Heidelberg Institute for Theoretical Studies
HITS

Motivation

- **coreference resolution**: which mentions in a text are used to refer to the same entity?

  Manchester United *have confirmed they are on the verge of signing* ***Bayern Munich*** ***midfielder Bastian Schweinsteiger****.*

  The World Cup winner *is set to undergo a medical and it is unlikely there will be problems agreeing personal terms with* ***him****.*

- tackled by increasingly complex structured prediction approaches
- key idea: **view coreference resolution as prediction of latent structures**

Toolkit Overview

- Python implementation of framework proposed in **Martschat and Strube (TACL, 2015)**, extending analysis toolkit **cort**
- **graph-based representation** of structures
- learning: **structured perceptron with cost-augmented inference**
- already implemented: mention pair, mention ranking, antecedent trees
- works on CoNLL data and raw text
- **source code and more:**

  `http://github.com/smartschat/cort`
- **install:**

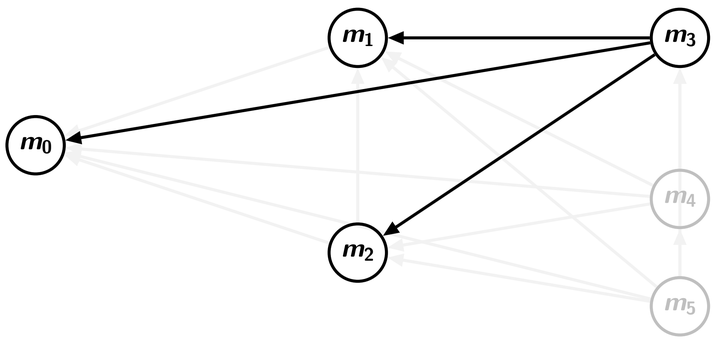
  `pip install cort`

Plug-and-Play Coreference Resolution

Example: Mention Ranking Model

- cast coreference as ranking of candidate antecedents for anaphor

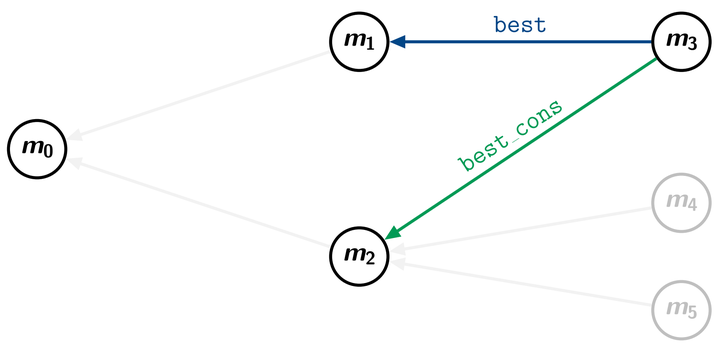
Define search space for latent (sub)structures




```
def extract_substructures(doc):
  substructures = []

  # iterate over mentions
  for i, ana in enumerate(doc.system_mentions):
      for_anaphor_arcs = []

      # iterate in reversed order over candidate antecedents
      for ante in sorted(doc.system_mentions[:i], reverse=True):
          for_anaphor_arcs.append((ana, ante))

      substructures.append(for_anaphor_arcs)

  return substructures
```

- represent each document as a graph
- instances: substructures (colored black)
- define valid arcs for substructures

Define a decoder

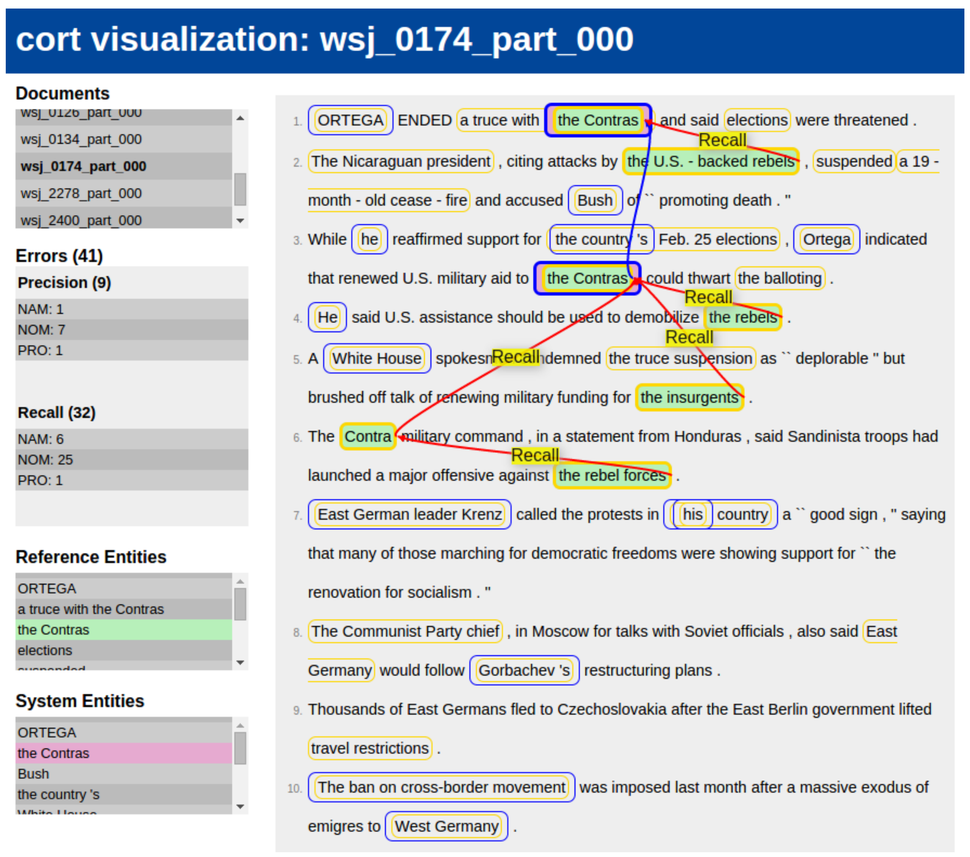



```
class RankingPerceptron(perceptrons.Perceptron):
    def argmax(self, substructure, arc_information):
        (best, max_val, best_cons, max_cons, best_is_cons) = \
            self.find_best_arcs(substructure, arc_information)

        return ([best], [], [max_val], [best_cons], [],
                [max_cons], best_is_cons)
```

- get highest scoring structure & highest scoring structure consistent with gold
- supports incorporating cost functions

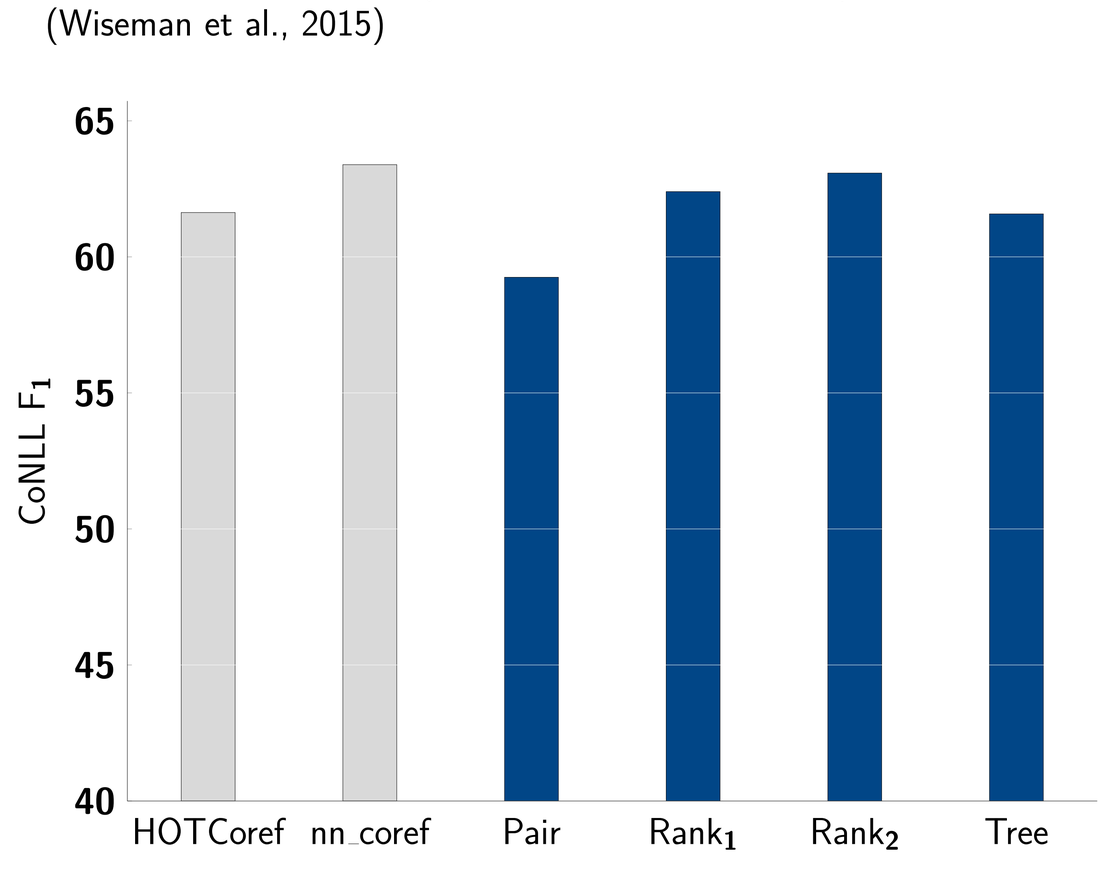
Visualization and Analysis

- building upon **error analysis toolkit** (Martschat et al., 2015)
- visualize and analyze decisions and errors of approaches



Results

- **state-of-the-art** performance on CoNLL-2012 data
- compare with HOTCoref (Björkelund and Kuhn, 2014) and nn_coref (Wiseman et al., 2015)



This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany.