Plug Latent Structures and Play Coreference Resolution

Sebastian Martschat, Patrick Claus and **Michael Strube**
Heidelberg Institute for Theoretical Studies gGmbH
Schloss-Wolfsbrunnenweg 35
69118 Heidelberg, Germany
(sebastian.martschat|patrick.claus|michael.strube)@h-its.org

Abstract

We present *cort*, a modular toolkit for devising, implementing, comparing and analyzing approaches to coreference resolution. The toolkit allows for a unified representation of popular coreference resolution approaches by making explicit the structures they operate on. Several of the implemented approaches achieve state-of-the-art performance.

1 Introduction

Coreference resolution is the task of determining which mentions in a text refer to the same entity. Machine learning approaches to coreference resolution range from simple binary classification models on mention pairs (Soon et al., 2001) to complex structured prediction approaches (Durrett and Klein, 2013; Fernandes et al., 2014).

In this paper, we present a toolkit that implements a framework that unifies these approaches: in the framework, we obtain a unified representation of many coreference approaches by making explicit the *latent structures* they operate on.

Our toolkit provides an interface for defining structures for coreference resolution, which we use to implement several popular approaches. An evaluation of the approaches on CoNLL shared task data (Pradhan et al., 2012) shows that they obtain state-of-the-art results. The toolkit also can perform end-to-end coreference resolution.

We implemented this functionality on top of the coreference resolution error analysis toolkit *cort* (Martschat et al., 2015). Hence, this toolkit now provides functionality for devising, implementing, comparing and analyzing approaches to coreference resolution. *cort* is released as open source[1] and is available from the Python Package Index[2].

[1]http://smartschat.de/software

[2]http://pypi.python.org/pypi. Install it via pip install cort.

2 A Framework for Coreference Resolution

In this section we briefly describe a structured prediction framework for coreference resolution.

2.1 Motivation

The popular mention pair approach (Soon et al., 2001; Ng and Cardie, 2002) operates on a *list* of mention pairs. Each mention pair is considered individually for learning and prediction. In contrast, antecedent tree models (Yu and Joachims, 2009; Fernandes et al., 2014; Björkelund and Kuhn, 2014) operate on a *tree* which encodes all anaphor-antecedent decisions in a document.

Conceptually, both approaches have in common that the structures they employ are not annotated in the data (in coreference resolution, the annotation consists of a mapping of mentions to entity identifiers). Hence, we can view both approaches as instantiations of a generic structured prediction approach with latent variables.

2.2 Setting

Our aim is to learn a prediction function f that, given an input document $x \in \mathcal{X}$, predicts a pair $(h, z) \in \mathcal{H} \times \mathcal{Z}$. h is the (unobserved) latent structure encoding the coreference relations between mentions in x. z is the mapping of mentions to entity identifiers (which is observed in the training data). Usually, z is obtained from h by taking the transitive closure over coreference decisions encoded in h. $\mathcal{H}$ and $\mathcal{Z}$ are the spaces containing all such structures and mappings.

2.3 Representation

For a document $x \in \mathcal{X}$, we write $M_x = \{m_1, \ldots, m_n\}$ for the mentions in x. Following previous work (Chang et al., 2012; Fernandes et al., 2014), we make use of a *dummy mention* which we denote as m_0. If m_0 is predicted as the antecedent of a mention m_i, we consider m_i non-anaphoric. We define $M_x^0 = \{m_0\} \cup M_x$.

Inspired by previous work (Bengtson and Roth, 2008; Fernandes et al., 2014; Martschat and Strube, 2014), we adopt a graph-based representation of the latent structures $h \in \mathcal{H}$. In particular, we express structures by *labeled directed graphs* with vertex set M_x^0.

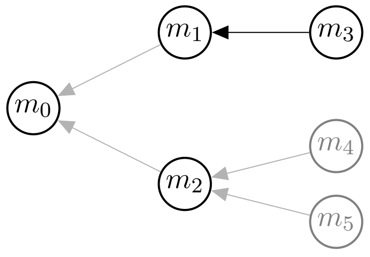


Figure 1: Latent structure underlying the mention ranking and the antecedent tree approach. The black nodes and arcs represent one substructure for the mention ranking approach.

Figure 1 shows a structure underlying the mention ranking and the antecedent tree approach. An arc between two mentions signals coreference. For antecedent trees (Fernandes et al., 2014), the whole structure is considered, while for mention ranking (Denis and Baldridge, 2008; Chang et al., 2012) only the antecedent decision for one anaphor is examined. This can be expressed via an appropriate segmentation into subgraphs which we refer to as *substructures*. One such substructure encoding the antecedent decision for m_3 is colored black in the figure.

Via arc labels we can express additional information. For example, mention pair models (Soon et al., 2001) distinguish between *positive* and *negative* instances. This can be modeled by labeling arcs with appropriate labels, such as $+$ and $-$.

2.4 Inference and Learning

As is common in natural language processing, we model the prediction of (h, z) via a linear model. That is,

$$f(x) = f_\theta(x) = \underset{(h,z) \in \mathcal{H} \times \mathcal{Z}}{\arg\max} \langle \theta, \phi(x, h, z) \rangle,$$

where $\theta \in \mathbb{R}^d$ is a parameter vector and $\phi\colon \mathcal{X} \times \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^d$ is a joint feature representation for inputs and outputs. When employing substructures, one maximization problem has to be solved for each substructure (instead of one maximization problem for the whole structure).

To learn the parameter vector $\theta \in \mathbb{R}^d$ from training data, we employ a latent structured perceptron (Sun et al., 2009) with cost-augmented inference (Crammer et al., 2006) and averaging (Collins, 2002).

3 Implementation

We now describe our implementation of the framework presented in the previous section.

3.1 Aims

By expressing approaches in the framework, researchers can quickly devise, implement, compare and analyze approaches for coreference resolution. To facilitate development, it should be as easy as possible to define a coreference resolution approach. We first describe the general architecture of our toolkit before giving a detailed description of how to implement specific coreference resolution approaches.

3.2 Architecture

The toolkit is implemented in Python. It can process raw text and data conforming to the format of the CoNLL-2012 shared task on coreference resolution (Pradhan et al., 2012). The toolkit is organized in four modules: the `preprocessing` module contains functionality for processing raw text, the `core` module provides mention extraction and computation of mention properties, the `analysis` module contains error analysis methods, and the `coreference` module implements the framework described in the previous section.

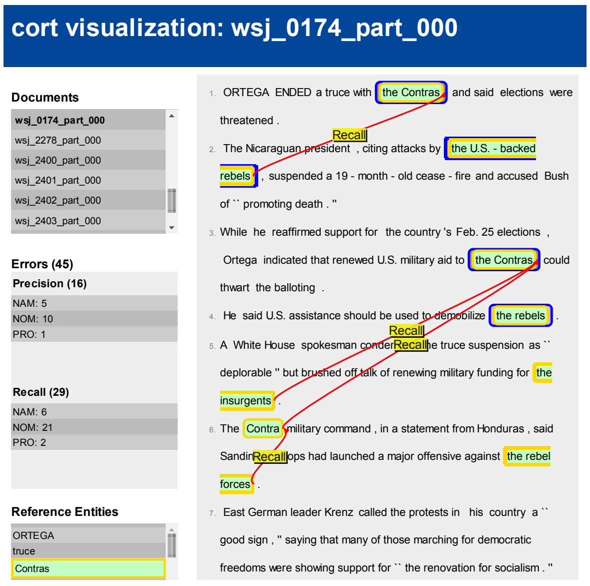
3.2.1 `preprocessing`

By making use of NLTK[3], this module provides classes and functions for performing the preprocessing tasks necessary for mention extraction and coreference resolution: tokenization, sentence splitting, parsing and named entity recognition.

3.2.2 `core`

We employ a rule-based mention extractor, which also computes a rich set of mention attributes, including tokens, head, part-of-speech tags, named entity tags, gender, number, semantic class, grammatical function and mention type. These attributes, from which features are computed, can be extended easily.

[3]`http://www.nltk.org/`


cort visualization: wsj_0174_part_000

Documents

wsj_0174_part_000
wsj_2278_part_000
wsj_2400_part_000
wsj_2401_part_000
wsj_2402_part_000
wsj_2403_part_000

Errors (45)

Precision (16)

NAM: 5
NOM: 10
PRO: 1

Recall (29)

NAM: 6
NOM: 21
PRO: 2

Reference Entities

ORTEGA
truce
Contras

1. ORTEGA ENDED a truce with the Contras and said elections were threatened .

2. The Nicaraguan president , citing attacks by the U.S. - backed rebels , suspended a 19 - month - old cease - fire and accused Bush of `` promoting death . ''

3. While he reaffirmed support for the country 's Feb. 25 elections , Ortega indicated that renewed U.S. military aid to the Contras could thwart the balloting .

4. He said U.S. assistance should be used to demobilize the rebels .

5. A White House spokesman conderRecallhe truce suspension as `` deplorable '' but brushed off talk of renewing military funding for the insurgents .

6. The Contra military command , in a statement from Honduras , said SandinRecallops had launched a major offensive against the rebel forces .

7. East German leader Krenz called the protests in his country a `` good sign , '' saying that many of those marching for democratic freedoms were showing support for `` the renovation for socialism . ''

Figure 2: Visualization of errors.

Listing 1 Instance extraction for the mention ranking model with latent antecedents.

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

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

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

    substructures.append(ana_arcs)

  return substructures
```

Listing 2 Decoder for the mention ranking model with latent antecedents.

```python
class RankingPerceptron(
    perceptrons.Perceptron):
  def argmax(self, substructure,
                 arc_information):
    best_arc, best_arc_score, \
    best_cons_arc, best_cons_arc_score, \
    consistent = self.find_best_arcs(
      substructure, arc_information)

    return ([best_arc], [],
            [best_arc_score],
            [best_cons_arc], [],
            [best_cons_arc_score],
            consistent)
```

3.2.3 `analysis`

To support system development, this module implements the error analysis framework of Martschat and Strube (2014). Users can extract, analyze and visualize recall and precision errors of the systems they are working on. Figure 2 shows a screenshot of the visualization. A more detailed description can be found in Martschat et al. (2015).

3.2.4 `coreference`

This module provides features for coreference resolution and implements the machine learning framework described in the previous section.

We implemented a rich set of features employed in previous work (Ng and Cardie, 2002; Bengtson and Roth, 2008; Björkelund and Kuhn, 2014), including lexical, rule-based and semantic features. The feature set can be extended by the user.

The module provides a structured latent perceptron implementation and contains classes that implement the workflows for training and prediction. As its main feature, it provides an interface for defining coreference resolution approaches. We already implemented various approaches (see Section 4).

3.3 Defining Approaches

The toolkit provides a simple interface for devising coreference resolution approaches via structures. The user just needs to specify two functions: an *instance extractor*, which defines the search space for the optimal (sub)structures, and a *decoder*, which, given a parameter vector, finds optimal (sub)structures. The toolkit then performs training and prediction using these user-specified functions. The user can further customize the approach by defining *cost functions* to be used during cost-augmented inference, and *clustering algorithms* to extract coreference chains from latent structures, such as closest-first (Soon et al., 2001) or best-first (Ng and Cardie, 2002).

In the remainder of this section, we present an example implementation of the mention ranking model with latent antecedents (Chang et al., 2012) in our toolkit.

3.3.1 Instance Extractors

The instance extractor receives a document as input and defines the search space for the maximization problem to be solved by the decoder. To do so, it needs to output the segmentation of the la-

Listing 3 Cost function for the mention ranking model with latent antecedents.

```
def cost_based_on_consistency(arc):
    ana, ante = arc

    consistent = \
      ana.decision_is_consistent(ante)

    # false new
    if not consistent and \
        ante.is_dummy():
      return 2
    # wrong link
    elif not consistent:
      return 1
    # correct
    else:
      return 0
```

tent structure for one document into substructures, and the candidate arcs for each substructure.

Listing 1 shows source code of the instance extractor for the mention ranking model with latent antecedents. In this model, each antecedent decision for a mention corresponds to one substructure. Therefore, the extractor iterates over all mentions. For each mention, arcs to all preceding mentions are extracted and stored as candidate arcs for one substructure.

3.3.2 Decoders

The decoder solves the maximization problems for obtaining the highest-scoring latent substructures consistent with the gold annotation, and the highest-scoring cost-augmented latent substructures.

Listing 2 shows source code of a decoder for the mention ranking model with latent antecedents. The input to the decoder is a *substructure*, which is a set of arcs, and a mapping from arcs to information about arcs, such as features or costs. The output is a tuple containing

- a list of arcs that constitute the highest-scoring substructure, together with their labels (if any) and scores,
- the same for the highest-scoring substructure consistent with the gold annotation,
- the information whether the highest-scoring substructure is consistent with the gold annotation.

To obtain this prediction, we invoke the auxiliary function `self.find_best_arcs`. This function searches through a set of arcs to find the overall highest-scoring arc and the overall highest-scoring arc consistent with the gold annotation. Furthermore, it also outputs the scores of these arcs according to the model, and whether the prediction of the best arc is consistent with the gold annotation.

For the mention ranking model, we let the function search through all candidate arcs for a substructure, since these represent the antecedent decision for one anaphor. Note that the mention ranking model does not use any labels.

The update of the parameter vector is handled by our implementation of the structured perceptron.

3.3.3 Cost Functions

Cost functions allow to bias the learner towards specific substructures, which leads to a large margin approach. For the mention ranking model, we employ a cost function that assigns a higher cost to erroneously determining anaphoricity than to selecting a wrong link, similar to the cost functions employed by Durrett and Klein (2013) and Fernandes et al. (2014). The source code is displayed in Listing 3.

3.3.4 Clustering Algorithms

The mention ranking model selects one antecedent for each anaphor, therefore there is no need to cluster antecedent decisions. Our toolkit provides clustering algorithms commonly used for mention pair models, such as *closest-first* (Soon et al., 2001) or *best-first* (Ng and Cardie, 2002).

3.4 Running *cort*

cort can be used as a Python library, but also provides two command line tools `cort-train` and `cort-predict`.

4 Evaluation

We implemented a mention pair model with best-first clustering (Ng and Cardie, 2002), the mention ranking model with closest (Denis and Baldridge, 2008) and latent (Chang et al., 2012) antecedents, and antecedent trees (Fernandes et al., 2014). Only slight modifications of the source code displayed in Listings 1 and 2 were necessary to implement these approaches. For the ranking models and antecedent trees we use the cost function described in Listing 3.

We evaluate the models on the English test data of the CoNLL-2012 shared task on multilingual coreference resolution (Pradhan et al., 2012). We use the reference implementation of the CoNLL

	MUC			B^3			$CEAF_e$			
Model	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**R**	**P**	**F_1**	**Average F_1**
CoNLL-2012 English test data										
Fernandes et al. (2014)	65.83	75.91	70.51	51.55	65.19	57.58	50.82	57.28	53.86	60.65
Björkelund and Kuhn (2014)	67.46	74.30	70.72	54.96	62.71	58.58	52.27	59.40	55.61	61.63
Mention Pair	67.16	71.48	69.25	51.97	60.55	55.93	51.02	51.89	51.45	58.88
Ranking: Closest	67.96	76.61	72.03	54.07	64.98	59.03	51.45	59.02	54.97	62.01
Ranking: Latent	68.13	76.72	72.17	54.22	66.12	59.58	52.33	59.47	55.67	62.47
Antecedent Trees	65.34	78.12	71.16	50.23	67.36	57.54	49.76	58.43	53.75	60.82

Table 1: Results of different systems and models on CoNLL-2012 English test data. Models below the dashed lines are implemented in our toolkit.

scorer (Pradhan et al., 2014), which computes the average of the evaluation metrics MUC (Vilain et al., 1995), B^3, (Bagga and Baldwin, 1998) and $CEAF_e$ (Luo, 2005). The models are trained on the concatenation of training and development data.

The evaluation of the models is shown in Table 1. To put the numbers into context, we compare with Fernandes et al. (2014), the winning system of the CoNLL-2012 shared task, and the state-of-the-art system of Björkelund and Kuhn (2014).

The mention pair model performs decently, while the antecedent tree model exhibits performance comparable to Fernandes et al. (2014), who use a very similar model. The ranking models outperform Björkelund and Kuhn (2014), obtaining state-of-the-art performance.

5 Related Work

Many researchers on coreference resolution release an implementation of the coreference model described in their paper (Lee et al., 2013; Durrett and Klein, 2013; Björkelund and Kuhn, 2014, inter alia). However, these implementations implement only one approach following one paradigm (such as mention ranking or antecedent trees).

Similarly to *cort*, research toolkits such as BART (Versley et al., 2008) or Reconcile (Stoyanov et al., 2009) provide a framework to implement and compare coreference resolution approaches. In contrast to these toolkits, we make the latent structure underlying coreference approaches explicit, which facilitates development of new approaches and renders the development more transparent. Furthermore, we provide a generic and customizable learning algorithm.

6 Conclusions

We presented an implementation of a framework for coreference resolution that represents approaches to coreference resolution by the structures they operate on. In the implementation we placed emphasis on facilitating the definition of new models in the framework.

The presented toolkit *cort* can process raw text and CoNLL shared task data. It achieves state-of-the-art performance on the shared task data.

The framework and toolkit presented in this paper help researchers to devise, analyze and compare representations for coreference resolution.

Acknowledgements

We thank Benjamin Heinzerling for helpful comments on drafts of this paper. This work has been funded by the Klaus Tschira Foundation, Germany. The first author has been supported by a HITS Ph.D. scholarship.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pages 563–566.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 294–303.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features.

In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Baltimore, Md., 22–27 June 2014, pages 47–57.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. Illinois-Coref: The UI system in the CoNLL-2012 shared task. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 113–117.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing,* Philadelphia, Penn., 6–7 July 2002, pages 1–8.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25–27 October 2008, pages 660–669.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 1971–1982.

Eraldo Fernandes, Cícero dos Santos, and Ruy Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics*, 40(4):801–835.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 25–32.

Sebastian Martschat and Michael Strube. 2014. Recall error analysis for coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 2070–2081.

Sebastian Martschat, Thierry Göckel, and Michael Strube. 2015. Analyzing and visualizing coreference resolution errors. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations,* Denver, Col., 31 May – 5 June 2015, pages 6–10.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* Philadelphia, Penn., 7–12 July 2002, pages 104–111.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 1–40.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Baltimore, Md., 22–27 June 2014, pages 30–35.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2009. Reconcile: A coreference resolution research platform. Technical report, Cornell University.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of the 21th International Joint Conference on Artificial Intelligence,* Pasadena, Cal., 14–17 July 2009, pages 1236–1242.

Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Companion Volume to the Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics,* Columbus, Ohio, 15–20 June 2008, pages 9–12.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, Cal. Morgan Kaufmann.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proceedings of the 26th International Conference on Machine Learning,* Montréal, Québec, Canada, 14–18 June 2009, pages 1169–1176.