Coreference Resolution with Reconcile

Veselin Stoyanov
Center for Language and Speech Processing
Johns Hopkins Univ.
Baltimore, MD
ves@cs.jhu.edu

Claire Cardie
Department of Computer Science
Cornell University
Ithaca, NY
cardie@cs.cornell.edu

Nathan Gilbert
Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT
ngilbert@cs.utah.edu
riloff@cs.utah.edu

David Buttler
David Hysom
Lawrence Livermore National Laboratory
Livermore, CA
buttler1@llnl.gov
hysom1@llnl.gov

Abstract

Despite the existence of several noun phrase coreference resolution data sets as well as several formal evaluations on the task, it remains frustratingly difficult to compare results across different coreference resolution systems. This is due to the high cost of implementing a complete end-to-end coreference resolution system, which often forces researchers to substitute available gold-standard information in lieu of implementing a module that would compute that information. Unfortunately, this leads to inconsistent and often unrealistic evaluation scenarios.

With the aim to facilitate consistent and realistic experimental evaluations in coreference resolution, we present Reconcile, an infrastructure for the development of learning-based noun phrase (NP) coreference resolution systems. Reconcile is designed to facilitate the rapid creation of coreference resolution systems, easy implementation of new feature sets and approaches to coreference resolution, and empirical evaluation of coreference resolvers across a variety of benchmark data sets and standard scoring metrics. We describe Reconcile and present experimental results showing that Reconcile can be used to create a coreference resolver that achieves performance comparable to state-of-the-art systems on six benchmark data sets.

1 Introduction

Noun phrase coreference resolution (or simply coreference resolution) is the problem of identifying all noun phrases (NPs) that refer to the same entity in a text. The problem of coreference resolution is fundamental in the field of natural language processing (NLP) because of its usefulness for other NLP tasks, as well as the theoretical interest in understanding the computational mechanisms involved in government, binding and linguistic reference.

Several formal evaluations have been conducted for the coreference resolution task (e.g., MUC-6 (1995), ACE NIST (2004)), and the data sets created for these evaluations have become standard benchmarks in the field (e.g., MUC and ACE data sets). However, it is still frustratingly difficult to compare results across different coreference resolution systems. Reported coreference resolution scores vary wildly across data sets, evaluation metrics, and system configurations.

We believe that one root cause of these disparities is the high cost of implementing an end-to-end coreference resolution system. Coreference resolution is a complex problem, and successful systems must tackle a variety of non-trivial subproblems that are central to the coreference task — e.g., mention/markable detection, anaphor identification — and that require substantial implementation efforts. As a result, many researchers exploit gold-standard annotations, when available, as a substitute for component technologies to solve these subproblems. For example, many published research results use gold standard annotations to identify NPs (substituting for mention/markable detection), to distinguish anaphoric NPs from non-anaphoric NPs (substituting for anaphoricity determination), to identify named entities (substituting for named entity recognition), and to identify the semantic types of NPs (substituting for semantic class identification). Unfortunately, the use of gold standard annotations for key/critical component technologies leads to an unrealistic evaluation setting, and makes it impossible to directly compare results against coreference resolvers that solve all of these subproblems from scratch.

Comparison of coreference resolvers is further hindered by the use of several competing (and non-trivial) evaluation measures, and data sets that have substantially different task definitions and annotation formats. Additionally, coreference resolution is a pervasive problem in NLP and many NLP applications could benefit from an effective coreference resolver that can be easily configured and customized.

To address these issues, we have created a platform for coreference resolution, called Reconcile, that can serve as a software infrastructure to support the creation of, experimentation with, and evaluation of coreference resolvers. Reconcile was designed with the following seven desiderata in mind:

- implement the basic underlying software architecture of contemporary state-of-the-art learning-based coreference resolution systems;

- support experimentation on most of the standard coreference resolution data sets;
- implement most popular coreference resolution scoring metrics;
- exhibit state-of-the-art coreference resolution performance (i.e., it can be configured to create a resolver that achieves performance close to the best reported results);
- can be easily extended with new methods and features;
- is relatively fast and easy to configure and run;
- has a set of pre-built resolvers that can be used as black-box coreference resolution systems.

While several other coreference resolution systems are publicly available (e.g., Poesio and Kabadjov (2004), Qiu et al. (2004) and Versley et al. (2008)), none meets all seven of these desiderata (see Related Work). Reconcile is a modular software platform that abstracts the basic architecture of most contemporary supervised learning-based coreference resolution systems (e.g., Soon et al. (2001), Ng and Cardie (2002), Bengtson and Roth (2008)) and achieves performance comparable to the state-of-the-art on several benchmark data sets. Additionally, Reconcile can be easily reconfigured to use different algorithms, features, preprocessing elements, evaluation settings and metrics.

In the rest of this paper, we review related work (Section 2), describe Reconcile's organization and components (Section 3) and show experimental results for Reconcile on six data sets and two evaluation metrics (Section 4).

2 Related Work

Several coreference resolution systems are currently publicly available. JavaRap (Qiu et al., 2004) is an implementation of the Lappin and Leass' (1994) Resolution of Anaphora Procedure (RAP). JavaRap resolves only pronouns and, thus, it is not directly comparable to Reconcile. GuiTaR (Poesio and Kabadjov, 2004) and BART (Versley et al., 2008) (which can be considered a successor of GuiTaR) are both modular systems that target the full coreference resolution task. As such, both systems come close to meeting the majority of the desiderata set forth in Section 1. BART, in particular, can be considered an alternative to Reconcile, although we believe that Reconcile's approach is more flexible than BART's. In addition, the architecture and system components of Reconcile (including a comprehensive set of features that draw on the expertise of state-of-the-art supervised learning approaches, such as Bengtson and Roth (2008)) result in performance closer to the state-of-the-art.

Coreference resolution has received much research attention, resulting in an array of approaches, algorithms and features. Reconcile is modeled after typical supervised learning approaches to coreference resolution (e.g. the architecture introduced by Soon et al. (2001)) because of the popularity and relatively good performance of these systems.

However, there have been other approaches to coreference resolution, including unsupervised and semi-supervised approaches (e.g. Haghighi and Klein (2007)), structured approaches (e.g. McCallum and Wellner (2004) and Finley and Joachims (2005)), competition approaches (e.g. Yang et al. (2003)) and a bell-tree search approach (Luo et al. (2004)). Most of these approaches rely on some notion of pairwise feature-based similarity and can be directly implemented in Reconcile.

3 System Description

Reconcile was designed to be a research testbed capable of implementing most current approaches to coreference resolution. Reconcile is written in Java, to be portable across platforms, and was designed to be easily reconfigurable with respect to subcomponents, feature sets, parameter settings, etc.

Reconcile's architecture is illustrated in Figure 1. For simplicity, Figure 1 shows Reconcile's operation during the classification phase (i.e., assuming that a trained classifier is present).

The basic architecture of the system includes five major steps. Starting with a corpus of documents together with a manually annotated coreference resolution answer key[1], Reconcile performs

[1]Only required during training.

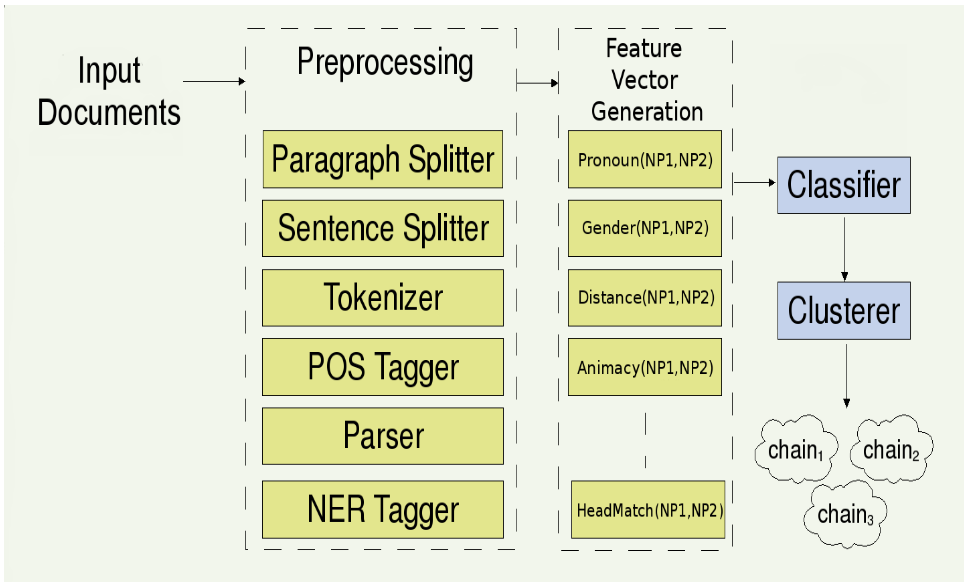

Figure 1: The Reconcile classification architecture.

the following steps, in order:

1. **Preprocessing.** All documents are passed through a series of (external) linguistic processors such as tokenizers, part-of-speech taggers, syntactic parsers, etc. These components produce annotations of the text. Table 1 lists the preprocessors currently interfaced in Reconcile. Note that Reconcile includes several in-house NP detectors, that conform to the different data sets' definitions of what constitutes a NP (e.g., MUC vs. ACE). All of the extractors utilize a syntactic parse of the text and the output of a Named Entity (NE) extractor, but extract different constructs as specialized in the corresponding definition. The NP extractors successfully recognize about 95% of the NPs in the MUC and ACE gold standards.

2. **Feature generation.** Using annotations produced during preprocessing, Reconcile produces feature vectors for pairs of NPs. For example, a feature might denote whether the two NPs agree in number, or whether they have any words in common. Reconcile includes over 80 features, inspired by other successful coreference resolution systems such as Soon et al. (2001) and Ng and Cardie (2002).

3. **Classification.** Reconcile learns a classifier that operates on feature vectors representing pairs of NPs and it is trained to assign a score indicating the likelihood that the NPs in the pair are coreferent.

4. **Clustering.** A clustering algorithm consolidates the predictions output by the classifier and forms the final set of coreference clusters (chains).[2]

5. **Scoring.** Finally, during testing Reconcile runs scoring algorithms that compare the chains produced by the system to the gold-standard chains in the answer key.

Task	Systems
Sentence splitter	UIUC (CC Group, 2009) OpenNLP (Baldridge, J., 2005)
Tokenizer	OpenNLP (Baldridge, J., 2005)
POS Tagger	OpenNLP (Baldridge, J., 2005) + the two parsers below
Parser	Stanford (Klein and Manning, 2003) Berkeley (Petrov and Klein, 2007)
Dep. parser	Stanford (Klein and Manning, 2003)
NE Recognizer	OpenNLP (Baldridge, J., 2005) Stanford (Finkel et al., 2005)
NP Detector	In-house

Table 1: Preprocessing components available in Reconcile.

Each of the five steps above can invoke different components. Reconcile's modularity makes it

[2]Some structured coreference resolution algorithms (e.g., McCallum and Wellner (2004) and Finley and Joachims (2005)) combine the classification and clustering steps above. Reconcile can easily accommodate this modification.

Step	Available modules
Classification	various learners in the Weka toolkit libSVM (Chang and Lin, 2001) SVM$_{light}$ (Joachims, 2002)
Clustering	Single-link Best-First Most Recent First
Scoring	MUC score (Vilain et al., 1995) B^3 score (Bagga and Baldwin, 1998) CEAF score (Luo, 2005)

Table 2: Available implementations for different modules available in Reconcile.

easy for new components to be implemented and existing ones to be removed or replaced. Reconcile's standard distribution comes with a comprehensive set of implemented components – those available for steps 2–5 are shown in Table 2. Reconcile contains over 38,000 lines of original Java code. Only about 15% of the code is concerned with running existing components in the preprocessing step, while the rest deals with NP extraction, implementations of features, clustering algorithms and scorers. More details about Reconcile's architecture and available components and features can be found in Stoyanov et al. (2010).

4 Evaluation

4.1 Data Sets

Reconcile incorporates the six most commonly used coreference resolution data sets, two from the MUC conferences (MUC-6, 1995; MUC-7, 1997) and four from the ACE Program (NIST, 2004). For ACE, we incorporate only the newswire portion. When available, Reconcile employs the standard test/train split. Otherwise, we randomly split the data into a training and test set following a 70/30 ratio. Performance is evaluated according to the B^3 and MUC scoring metrics.

4.2 The $Reconcile_{2010}$ Configuration

Reconcile can be easily configured with different algorithms for markable detection, anaphoricity determination, feature extraction, etc., and run against several scoring metrics. For the purpose of this sample evaluation, we create only one particular instantiation of Reconcile, which we will call $Reconcile_{2010}$ to differentiate it from the general platform. $Reconcile_{2010}$ is configured using the following components:

1. **Preprocessing**
   (a) **Sentence Splitter:** *OpenNLP*
   (b) **Tokenizer:** *OpenNLP*
   (c) **POS Tagger:** *OpenNLP*
   (d) **Parser:** *Berkeley*
   (e) **Named Entity Recognizer:** *Stanford*
2. **Feature Set** - A hand-selected subset of 60 out of the more than 80 features available. The features were selected to include most of the features from Soon et al. Soon et al. (2001), Ng and Cardie (2002) and Bengtson and Roth (2008).
3. **Classifier** - *Averaged Perceptron*
4. **Clustering** - *Single-link* - Positive decision threshold was tuned by cross validation of the training set.

4.3 Experimental Results

The first two rows of Table 3 show the performance of $Reconcile_{2010}$. For all data sets, B^3 scores are higher than MUC scores. The MUC score is highest for the MUC6 data set, while B^3 scores are higher for the ACE data sets as compared to the MUC data sets.

Due to the difficulties outlined in Section 1, results for Reconcile presented here are directly comparable only to a limited number of scores reported in the literature. The bottom three rows of Table 3 list these comparable scores, which show that $Reconcile_{2010}$ exhibits state-of-the-art performance for supervised learning-based coreference resolvers. A more detailed study of Reconcile-based coreference resolution systems in different evaluation scenarios can be found in Stoyanov et al. (2009).

5 Conclusions

Reconcile is a general architecture for coreference resolution that can be used to easily create various coreference resolvers. Reconcile provides broad support for experimentation in coreference resolution, including implementation of the basic architecture of contemporary state-of-the-art coreference systems and a variety of individual modules employed in these systems. Additionally, Reconcile handles all of the formatting and scoring peculiarities of the most widely used coreference resolution data sets (those created as part of the MUC and ACE conferences) and, thus, allows for easy implementation and evaluation across these data sets. We hope that Reconcile will support experimental research in coreference resolution and provide a state-of-the-art coreference resolver for both researchers and application developers. We believe that in this way Reconcile will facilitate meaningful and consistent comparisons of coreference resolution systems. The full Reconcile release is available for download at `http://www.cs.utah.edu/nlp/reconcile/`.

System	Score	Data sets					
		MUC6	MUC7	ACE-2	ACE03	ACE04	ACE05
$Reconcile_{2010}$	MUC	68.50	62.80	65.99	67.87	62.03	67.41
	B^3	70.88	65.86	78.29	79.39	76.50	73.71
Soon et al. (2001)	MUC	62.6	60.4	–	–	–	–
Ng and Cardie (2002)	MUC	70.4	63.4	–	–	–	–
Yang et al. (2003)	MUC	71.3	60.2	–	–	–	–

Table 3: Scores for Reconcile on six data sets and scores for comparable coreference systems.

Acknowledgments

This research was supported in part by the National Science Foundation under Grant # 0937060 to the Computing Research Association for the CIFellows Project, Lawrence Livermore National Laboratory subcontract B573245, Department of Homeland Security Grant N0014-07-1-0152, and Air Force Contract FA8750-09-C-0172 under the DARPA Machine Reading Program.

The authors would like to thank the anonymous reviewers for their useful comments.

References

A. Bagga and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *Linguistic Coreference Workshop at the Language Resources and Evaluation Conference*.

Baldridge, J. 2005. The OpenNLP project. http://opennlp.sourceforge.net/.

E. Bengtson and D. Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

CC Group. 2009. Sentence Segmentation Tool. http://l2r.cs.uiuc.edu/ cogcomp/atool.php?tkey=SS.

C. Chang and C. Lin. 2001. LIBSVM: a Library for Support Vector Machines. Available at `http://www.csie.ntu.edu.tw/cjlin/libsvm`.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*.

T. Finley and T. Joachims. 2005. Supervised clustering with support vector machines. In *Proceedings of the Twenty-second International Conference on Machine Learning (ICML 2005)*.

A. Haghighi and D. Klein. 2007. Unsupervised Coreference Resolution in a Nonparametric Bayesian Model. In *Proceedings of the 45th Annual Meeting of the ACL*.

T. Joachims. 2002. SVM_{Light}, http://svmlight.joachims.org.

D. Klein and C. Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing (NIPS 2003)*.

S. Lappin and H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Annual Meeting of the ACL*.

X. Luo. 2005. On Coreference Resolution Performance Metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

A. McCallum and B. Wellner. 2004. Conditional Models of Identity Uncertainty with Application to Noun Coreference. In *Advances in Neural Information Processing (NIPS 2004)*.

MUC-6. 1995. Coreference Task Definition. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*.

MUC-7. 1997. Coreference Task Definition. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.

V. Ng and C. Cardie. 2002. Improving Machine Learning Approaches to Coreference Resolution. In *Proceedings of the 40th Annual Meeting of the ACL*.

NIST. 2004. *The ACE Evaluation Plan*. NIST.

S. Petrov and D. Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of the Joint Meeting of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2007)*.

M. Poesio and M. Kabadjov. 2004. A general-purpose, off-the-shelf anaphora resolution module: implementation and preliminary evaluation. In *Proceedings of the Language Resources and Evaluation Conference*.

L. Qiu, M.-Y. Kan, and T.-S. Chua. 2004. A public reference implementation of the rap anaphora resolution algorithm. In *Proceedings of the Language Resources and Evaluation Conference*.

W. Soon, H. Ng, and D. Lim. 2001. A Machine Learning Approach to Coreference of Noun Phrases. *Computational Linguistics*, 27(4):521–541.

V. Stoyanov, N. Gilbert, C. Cardie, and E. Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of ACL/IJCNLP*.

V. Stoyanov, C. Cardie, N. Gilbert, E. Riloff, D. Buttler, and D. Hysom. 2010. Reconcile: A coreference resolution research platform. Technical report, Cornell University.

Y. Versley, S. Ponzetto, M. Poesio, V. Eidelman, A. Jern, J. Smith, X. Yang, and A. Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Proceedings of the Language Resources and Evaluation Conference*.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A Model-Theoretic Coreference Scoring Theme. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*.

X. Yang, G. Zhou, J. Su, and C. Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the 41st Annual Meeting of the ACL*.