Reconciling OntoNotes: Unrestricted Coreference Resolution in OntoNotes with Reconcile

Veselin Stoyanov
CLSP
Johns Hopkins University
Baltimore, MD

Uday Babbar and **Pracheer Gupta** and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY

Abstract

This paper describes our entry to the 2011 CoNLL closed task (Pradhan et al., 2011) on modeling unrestricted coreference in OntoNotes. Our system is based on the Reconcile coreference resolution research platform. Reconcile is a general software infrastructure for the development of learning-based noun phrase (NP) coreference resolution systems. Our entry for the CoNLL closed task is a configuration of Reconcile intended to do well on OntoNotes data. This paper describes our configuration of Reconcile as well as the changes that we had to implement to integrate with the OntoNotes task definition and data formats. We also present and discuss the performance of our system under different testing conditions on a withheld validation set.

1 Introduction

Noun phrase (NP) coreference resolution is one of the fundamental tasks of the field of Natural Language Processing (NLP). Recently, the creation of the OntoNotes corpus (Pradhan et al., 2007) has provided researchers with a large standard data collection with which to create and empirically compare coreference resolution systems.

Reconcile (Stoyanov et al., 2010b) is a general coreference resolution research platform that aims to abstract the architecture of different learning-based coreference systems and to provide infrastructure for their quick implementation. Reconcile is distributed with several state-of-the art NLP components and a set of optimized feature implementations. We decided to adapt Reconcile for the OntoNotes corpus and enter it in the 2011 CoNLL shared task with three goals in mind: (i) to compare the architecture and components of Reconcile with other state-of-the-art coreference systems, (ii) to implement and provide the capability of running Reconcile on the OntoNotes corpus, and, (iii) to provide a baseline for future algorithm implementations in Reconcile that evaluate on the OntoNotes corpus.

Although Reconcile can be easily adapted to new corpora, doing so requires introducing new components. More precisely, the system has to be modified to be consistent with the specific definition of the coreference task embodied in the OntoNotes annotation instructions. Additionally, different corpora use different data formats, so the system needs to implement capabilities for dealing with these new formats. Finally, Reconcile can be configured with different features and components to create an instantiation that models well the particular data.

In this paper we describe, $Reconcile_{CoNLL}$, our entry to the 2011 CoNLL shared task based on the Reconcile research platform. We begin by describing the general Reconcile architecture (Section 2), then describe the changes that we incorporated in order to enable Reconcile to work on OntoNotes data (Sections 3 and 4). Finally, we describe our experimental set up and results from running $Reconcile_{CoNLL}$ under different conditions (Section 5).

2 Overview of Reconcile

In this section we give a high-level overview of the Reconcile platform. We refer the reader for more details to Stoyanov et al. (2010a) and Stoyanov et al. (2010b). Results from running a Reconcile-based coreference resolution system on different corpora can be found in Stoyanov et al. (2009).

Reconcile was developed to be a coreference resolution research platform that allows for quick implementation of coreference resolution systems. The platform abstracts the major processing steps (components) of current state-of-the-art learning-based coreference resolution systems. A description of the steps and the available components can be found in the referenced papers.

3 The $Reconcile_{CoNLL}$ System

To participate in the 2011 CoNLL shared task, we configured Reconcile to conform to the OntoNotes general coreference resolution task. We will use the name $Reconcile_{CoNLL}$, to refer to this particular instantiation of the general Reconcile platform. The remainder of this section describe the changes required to enable $Reconcile_{CoNLL}$ to run (accurately) on OntoNotes data.

$Reconcile_{CoNLL}$ employs the same basic pipelined architecture as Reconcile. We describe the specific components used in each step.

1. Preprocessing. Documents in the OntoNotes corpus are manually (or semi-automatically) annotated with many types of linguistic information. This information includes tokens, part-of-speech tags, and named entity information as well as a constituent syntactic parse of the text. For the purpose of participating in the shared task, we rely on these manual annotations, when available. Thus, we do not run most of the standard Reconcile preprocessing components. One type of information not provided in the OntoNotes corpus is a dependency parse. Several of Reconcile's features rely on a dependency parse of the text. Thus, we ran the Stanford dependency parser (Klein and Manning, 2003), which performs a constituent parse and uses rules to convert to a dependency format.[1]

Two additional changes to the preprocessing step were necessary for running on the OntoNotes data. The first is the implementation of components that can convert data from the OntoNotes format to the Reconcile internal format. The second is adaptation of the Coreference Element (CE) extractor to conform to the OntoNotes definition of what can constitute a CE. Our implementations for these two tasks are briefly described in Sections 4.1 and 4.2, respectively.

2. Feature generation. $Reconcile_{CoNLL}$ was configured with 61 features that have proven successful for coreference resolution on other data sets. Due to the lack of time we performed no feature engineering or selection specific to OntoNotes. We used a new component for generating the pairwise CEs that comprise training and test instances, which we dub SMARTPG (for smart pair generator). This is described in Section 4.3.

3. Classification. We train a linear classifier using the averaged perceptron algorithm (Freund and Schapire, 1999). We use a subset of 750 randomly selected documents for training, since training on the entire set required too much memory.[2] As a result, we had ample validation data for tuning thresholds, etc.

[1]A better approach would be to use the rules to create the dependency parse from the manual constituent parse. We decided against this approach due to implementation overhead.

[2]It is easy to address the memory issue in the on-line perceptron setting, but in the interest of time we chose to reduce the size of the training data. Training on the set of 750 documents is done efficiently in memory by allocating 4GB to the Java virtual machine.

4. Clustering. We use Reconcile's single-link clustering algorithm. In other words, we compute the transitive closure of the positive pairwise predictions. Note that what constitutes a positive prediction depends on a threshold set for the classifier from the previous step. This clustering threshold is optimized using validation data. More details about the influence of the validation process can be found in Section 5.

5. Scoring. The 2011 CoNLL shared task provides a scorer that computes a set of commonly used coreference resolution evaluation metrics. We report results using this scorer in Section 5. However, we used the Reconcile-internal versions of scorers to optimize the threshold. This was done for pragmatic reasons – time pressure prevented us from incorporating the CoNLL scorer in the system. We also report the Reconcile-internal scores in the experiment section.

This concludes the high-level description of the $Reconcile_{CoNLL}$ system. Next, we describe in more detail the main changes implemented to adapt to the OntoNotes data.

4 Adapting to OntoNotes

The first two subsection below describe the two main tasks that need to be addressed when running Reconcile on a new data set: annotation conversion and CE extraction. The third subsection describes the new Smart CE Pairwise instance generator — a general component that can be used for any coreference data set.

4.1 Annotation Conversion

There are fundamental differences between the annotation format used by OntoNotes and that used internally by Reconcile. While OntoNotes relies on token-based representations, Reconcile uses a stand-off bytespan annotation. A significant part of the development of $Reconcile_{CoNLL}$ was devoted to conversion of the OntoNotes manual token, parse, named-entity and coreference annotations. In general, we prefer the stand-off bytespan format because it allows the reference text of the document to remain unchanged while annotation layers are added as needed.

4.2 Coreference Element Extraction

The definition of what can constitute an element participating in the coreference relation (i.e., a Coreference Element or CE) depends on the particular dataset. Optimizing the CE extraction com-

Optimized Metric	Thres-hold	B-Cubed	CEAF	MUC
BCubed	0.4470	**0.7112**	0.1622	0.6094
CEAF	0.4542	0.7054	**0.1650**	0.6141
MUC	0.4578	0.7031	0.1638	**0.6148**

Table 1: Reconcile-internal scores for different thresholds. The table lists the best threshold for the validation data and results using that threshold.

Pair Gen.	BCubed	CEAFe	MUC
SMARTPG	0.6993	0.1634	0.6126
All Pairs	0.6990	0.1603	0.6095

Table 3: Influence of different pair generators.

ponent for the particular task definition can result in dramatic improvements in performance. An accurate implementation limits the number of elements that the coreference system needs to consider while keeping the recall high.

The CE extractor that we implemented for OntoNotes extends the existing Reconcile ACE05 CE extractor (ACE05, 2005) via the following modifications:

Named Entities: We exclude named entities of type CARDINAL NUMBER, MONEY and NORP, the latter of which captures nationality, religion, political and other entities.

Possessives: In the OntoNotes corpus, possessives are included as coreference elements, while in ACE they are not.

$Reconcile_{CoNLL}$ ignores the fact that verbs can also be CEs for the OntoNotes coreference task as this change would have constituted a significant implementation effort.

Overall, our CE extractor achieves recall of over 96%, extracting roughly twice the number of CEs in the answer key (precision is about 50%). High recall is desirable for the CE extractor at the cost of precision since the job of the coreference system is to further narrow down the set of anaphoric CEs.

4.3 Smart Pair Generator

Like most current coreference resolution systems, at the heart of Reconcile lies a pairwise classifier. The job of the classifier is to decide whether or not two CEs are coreferent or not. We use the term *pair generation* to refer to the process of creating the CE pairs that the classifier considers. The most straightforward way of generating pairs is by enumerating all possible unique combinations. This approach has two undesirable properties – it requires time in the order of $O(n^2)$ for a given document (where n is the number of CEs in the document) and it produces highly imbalanced data sets with the number of positive instances (i.e., coreferent CEs) being a small fraction of the number of negative instances. The latter issue has been addressed by a technique named instance generation (Soon et al., 2001): during training, each CE is matched with the first preceding CE with which it corefers and all other CEs that reside in between the two. During testing, a CE is compared to all preceding CEs until a coreferent CE is found or the beginning of the document is reached. This technique reduces class imbalance, but it has the same worst-case runtime complexity of $O(n^2)$.

We employ a new type of pair generation that aims to address both the class imbalance and improves the worst-case runtime. We will use SMARTPG to refer to this component. Our pair generator relies on linguistic intuitions and is based on the type of each CE. For a given CE, we use a rule-based algorithm to guess its type. Based on the type, we restrict the scope of possible antecedents to which the CE can refer in the following way:

Proper Name (Named Entity): A proper name is compared against all proper names in the 20 preceding sentences. In addition, it is compared to all other CEs in the two preceding sentences.

Definite noun phrase: Compared to all CEs in the six preceding sentences.

Common noun phrase: Compared to all CEs in the two preceding sentences.

Pronoun: Compared to all CEs in the two preceding sentences unless it is a first person pronoun. First person pronouns are additionally compared to first person pronouns in the preceding 20 sentences.

During development, we used SMARTPG on coreference resolution corpora other than OntoNotes and determined that the pair generator tends to lead to more accurate results. It also has runtime linear in the number of CEs in a document, which leads to a sizable reduction in running time for large documents. Training files generated by SMARTPG also tend to be more balanced. Finally, by omitting pairs that are unlikely to be coreferent, SMARTPG produces much smaller training sets. This leads to faster learning and allows us to train on more documents.

Optimized Metric	Threshold	BCubed	CEAFe	MUC	BLANC	CEAFm	**Combined**
BCubed	0.4470	0.6651	0.4134	0.6156	0.6581	0.5249	0.5647
CEAF	0.4542	0.6886	0.4336	0.6206	0.7012	0.5512	0.5809
MUC	0.4578	**0.6938**	**0.4353**	**0.6215**	**0.7108**	**0.5552**	**0.5835**

Table 2: CoNLL scores for different thresholds on **validation data**.

CoNLL Official Test Scores	BCubed	CEAFe	MUC	BLANC	CEAFm	**Combined**
Closed Task	0.6144	0.3588	0.5843	0.6088	0.4608	**0.5192**
Gold Mentions	0.6248	0.3664	0.6154	0.6296	0.4808	**0.5355**

Table 4: Official CoNLL 2011 test scores. Combined score is the average of MUC, BCubed and CEAFe.

5 Experiments

In this section we present and discuss the results for $Reconcile_{CoNLL}$when trained and evaluated on OntoNotes data. For all experiments, we train on a set of 750 randomly selected documents from the OntoNotes corpus. We use another 674 randomly selected documents for validation. We report scores using the scorers implemented internally in Reconcile as well as the scorers supplied by the CoNLL shared task.

In the rest of the section, we describe our results when controlling two aspects of the system – the threshold of the pairwise CE classifier, which is tuned on training data, and the method used for pair generation. We conclude by presenting the official results for the CoNLL shared task.

Influence of Classifier Threshold As previously mentioned, the threshold above which the decision of the classifier is considered positive provides us with a knob that controls the precision/recall trade-off. Reconcile includes a module that can automatically search for a threshold value that optimizes a particular evaluation metric. Results using three Reconcile-internal scorers (BCubed, CEAF, MUC) are shown in Table 1. First, we see that the threshold that optimizes performance on the validation data also exhibits the best results on the test data. The same does not hold when using the CoNLL scorer for testing, however: as Table 2 shows, the best results for almost all of the CoNLL scores are achieved at the threshold that optimizes the Reconcile-internal MUC score. Note that we did not optimize thresholds for the external scorer in the name of saving implementation effort. Unfortunately, the results that we submitted for the official evaluations were for the suboptimal threshold that optimizes Reconcile-internal BCubed score.

Influence of Pair Generation Strategies Next, we evaluate the performance of SMARTPG pair generators. We run the same system set-up as above substituting the pair generation module. Results (using the internal scorer), displayed in Table 3, show our SMARTPG performs identically to the generator producing all pairs, while it runs in time linear in the number of CEs.

Official Scores for the CoNLL 2011 Shared Task Table 4 summarizes the official scores of $Reconcile_{CoNLL}$ on the CoNLL shared task. Surprisingly, the scores are substationally lower than the scores on our held-out training set. So far, we have no explanation for these differences in performance. We also observe that using gold-standard instead of system-extracted CEs leads to improvement in score of about point and a half.

The official score places us 8th out of 21 systems on the closed task. We note that because of the threshold optimization mix-up we suffered about 2 points in combined score performance. Realistically our system should score around 0.54 placing us 5th or 6th on the task.

6 Conclusions

In this paper, we presented $Reconcile_{CoNLL}$, our system for the 2011 CoNLL shared task based on the Reconcile research platform. We described the overall Reconcile platform, our configuration for the CoNLL task and the changes that we implemented specific to the task. We presented the results of an empirical evaluation performed on held-out training data. We discovered that results for our system on this data are quite different from the official score that our system achieved.

Acknowledgments
This material is based upon work supported by the National Science Foundation under Grant # 0937060 to the Computing Research Association for the CIFellows Project.

References

ACE05. 2005. NIST ACE evaluation website. In *http://www.nist.gov/speech/tests/ace/2005*.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. In *Machine Learning*, pages 277–296.

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)*.

Sameer S. Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in ontonotes. In *Proceedings of the International Conference on Semantic Computing*.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL 2011)*, Portland, Oregon, June.

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. 2010a. Reconcile: A coreference resolution research platform. Technical report, Cornell University.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010b. Coreference resolution with reconcile. In *Proceedings of the ACL 2010*.