Coreference for Learning to Extract Relations: Yes, Virginia, Coreference Matters

Ryan Gabbard rgabbard@bbn.com

Marjorie Freedman mfreedma@bbn.com

Ralph Weischedel weischedel@bbn.com

Raytheon BBN Technologies, 10 Moulton St., Cambridge, MA 02138

Abstract

As an alternative to requiring substantial supervised relation training data, many have explored bootstrapping relation extraction from a few seed examples. Most techniques assume that the examples are based on easily spotted anchors, e.g., names or dates. Sentences in a corpus which contain the anchors are then used to induce alternative ways of expressing the relation. We explore whether coreference can improve the learning process. That is, if the algorithm considered examples such as *his sister*, would accuracy be improved? With coreference, we see on average a 2-fold increase in F-Score. Despite using potentially errorful machine coreference, we see significant increase in recall on all relations. Precision increases in four cases and decreases in six.

1 Introduction

As an alternative to requiring substantial supervised relation training data (e.g. the ~300k words of detailed, exhaustive annotation in Automatic Content Extraction (ACE) evaluations[1]) many have explored bootstrapping relation extraction from a few (~20) seed instances of a relation. Key to such approaches is a large body of unannotated text that can be iteratively processed as follows:

1. Find sentences containing the seed instances.
2. Induce patterns of context from the sentences.
3. From those patterns, find more instances.
4. Go to 2 until some condition is reached.

Most techniques assume that **relation instances**, like hasBirthDate(Wolfgang Amadeus Mozart, 1756), are realized in the corpus as **relation texts**[2] with easily spotted anchors like *Wolfgang Amadeus Mozart was born in 1756.*

In this paper we explore whether using coreference can improve the learning process. That is, if the algorithm considered texts like *his birth in 1756* for the above relation, would performance of the learned patterns be better?

2 Related Research

There has been much work in relation extraction both in traditional supervised settings and, more recently, in bootstrapped, semi-supervised settings. To set the stage for discussing related work, we highlight some aspects of our system. Our work initializes learning with about 20 seed relation instances and uses about 9 million documents of unannotated text[3] as a background bootstrapping corpus. We use both normalized syntactic structure and surface strings as features.

Much has been published on learning relation extractors using lots of supervised training, as in ACE, which evaluates system performance in detecting a fixed set of concepts and relations in text. Researchers have typically used this data to incorporate a great deal of structural syntactic information in their models (e.g. Ramshaw, 2001), but the obvious weakness of these approaches is the resulting reliance on manually annotated examples, which are expensive and time-consuming to create.

[1] http://www.nist.gov/speech/tests/ace/

[2] Throughout we will use **relation instance** to refer to a fact (e.g. *ORGHasEmployee(Apple, Steve Jobs)*), while we will use **relation text** to refer a particular sentence entailing a relation instance (e.g. *Steve Jobs is Apple's CEO*).

[3] Wikipedia and the LDC's Gigaword newswire corpus.

Others have explored automatic pattern generation from seed examples. Agichtein & Gravano (2000) and Ravichandran & Hovy (2002) reported results for generating surface patterns for relation identification; others have explored similar approaches (e.g. Pantel & Pennacchiotti, 2006). Mitchell et al. (2009) showed that for macro-reading, precision and recall can be improved by learning a large set of interconnected relations and concepts simultaneously. In all cases, the approaches used surface (word) patterns without coreference. In contrast, we use the structural features of predicate-argument structure and employ coreference. Section 3 describes our particular approach to pattern and relation instance scoring and selection.

Another research strand (Chen et al., 2006 & Zhou et al., 2008) explores semi-supervised relation learning using the ACE corpus and assuming manual mention markup. They measure the accuracy of relation extraction alone, without including the added challenge of resolving non-specific relation arguments to name references. They limit their studies to the small ACE corpora where mention markup is manually encoded.

Most approaches to automatic pattern generation have focused on precision, e.g., Ravichandran and Hovy (2002) report results in the Text Retrieval Conference (TREC) Question Answering track, where extracting one text of a relation instance can be sufficient, rather than detecting all texts. Mitchell et al. (2009), while demonstrating high precision, do not measure recall.

In contrast, our study has emphasized recall. A primary focus on precision allows one to ignore many relation texts that require coreference or long-distance dependencies; one primary goal of our work is to measure system performance in exactly those areas. There are at least two reasons to not lose sight of recall. For the majority of entities there will be only a few mentions of that entity in even a large corpus. Furthermore, for many information-extraction problems the number documents at runtime will be far less than web-scale.

3 Approach

Figure 1 depicts our approach for learning patterns to detect relations. At each iteration, the steps are:

(1) Given the current relation instances, find possible texts that entail the relation by finding sentences in the corpus containing all arguments of an instance.

(2) As in Freedman et al. (2010) and Boschee et al. (2008), induce possible patterns using the context in which the arguments appear. Patterns include both surface strings and normalized syntactic structures.[4] Each proposed pattern is applied to the corpus to find a set of hypothesized texts. For each pattern, a confidence score is assigned using estimated precision[5] and recall. The highest confidence patterns are added to the pattern set.[6]

(3) The patterns are applied to the corpus to find additional possible relation instances. For each proposed instance, we estimate a score using a Naive Bayes model with the patterns as the features. When using coreference, this score is penalized if an instance's supporting evidence involves low-confidence coreference links. The highest scoring instances are added to the instance set.

(4) After the desired number of iterations (in these experiments, 20) is complete, a human reviews the resulting pattern set and removes those patterns which are clearly incorrect (e.g. '*X visited Y*' for *hasBirthPlace*).[7]

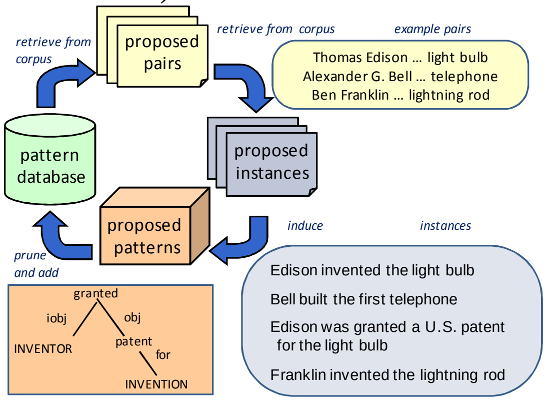
retrieve from corpus
proposed pairs
retrieve from corpus
example pairs
Thomas Edison … light bulb
Alexander G. Bell … telephone
Ben Franklin … lightning rod
pattern database
proposed instances
proposed patterns
induce
instances
prune and add
granted
iobj
obj
INVENTOR
patent
for
INVENTION
Edison invented the light bulb
Bell built the first telephone
Edison was granted a U.S. patent for the light bulb
Franklin invented the lightning rod

Figure 1: Approach to learning relations

We ran this system in two versions: –Coref has no access to coreference information, while +Coref (the original system) does. The systems are otherwise identical. Coreference information is provided by BBN's state-of-the-art information extraction

[4] Surface text patterns with wild cards are not proposed until the third iteration.

[5] Estimated recall is the weighted fraction of known instances found. Estimated precision is the weighted average of the scores of matched instances; scores for unseen instances are 0.

[6] As more patterns are accepted in a given iteration, we raise the confidence threshold. Usually, ~10 patterns are accepted per iteration.

[7] This takes about ten minutes per relation, which is less than the time to choose the initial seed instances.

system (Ramshaw, et al., 2011; NIST, 2007) in a mode which sacrifices some accuracy for speed (most notably by reducing the parser's search space). The IE system processes over 50MB/hour with an average EDR Value score when evaluated on an 8-fold cross-validation of the ACE 2007.

+Coref can propose relation instances from text in which the arguments are expressed as either name or non-name mentions. When the text of an argument of a proposed instance is a non-name, the system uses coreference to resolve the non-name to a name. -Coref can only propose instances based on texts where both arguments are names.[8]

This has several implications: If a text that entails a relation instance expresses one of the arguments as a non-name mention (e.g. *"Sue's husband is here."*), -Coref will be unable to learn an instance from that text. Even when all arguments are expressed as names, -Coref may need to use more specific, complex patterns to learn the instance (e.g. *"Sue asked her son, Bob, to set the table"*). We expect the ability to run using a 'denser,' more local space of patterns to be a significant advantage of +Coref. Certain types of patterns (e.g. patterns involving possessives) may also be less likely to be learned by -Coref. Finally, +Coref has access to much more training data at the outset because it can find more matching seed instances,[9] potentially leading to better and more stable training.

4 Evaluation Framework

Estimating recall for bootstrapped relation learning is a challenge except for corpora small enough for complete annotation to be feasible, e.g., the ACE corpora. ACE typically had a test set of ~30,000 words and ~300k for training. Yet, with a small corpus, rare relations will be inadequately represented.[10] Macro-reading evaluations (e.g. Mitchell, 2009) have not estimated recall, but have measured precision by sampling system output and determining whether the extracted fact is true in the world.

Ethel Kennedy says that when the family gathered for Thanksgiving she wanted the children to know what a real turkey looked like. So she sent ***her son****, Robert F. Kennedy Jr., to a farm to buy two birds.*

Figure 2: Passage entailing hasChild relation

Here we extend this idea to both precision and recall in a micro-reading context.

Precision is measured by running the system over the background corpus and randomly sampleing 100 texts that the system believes entail each relation. From the mentions matching the argument slots of the patterns, we build a relation instance. If these mentions are not names (only possible for +Coref), they are resolved to names using system coreference. For example, given the passage in Figure 2 and the pattern '(Y, poss:X)', the system would match the mentions X=her and Y=son, and build the relation instance *hasChild(Ethel Kennedy, Robert F. Kennedy Jr.).*

During assessment, the annotator is asked whether, in the context of the whole document, a given sentence entails the relation instance. We thus treat both incorrect relation extraction and incorrect reference resolution as mistakes.

To measure recall, we select 20 test relation instances and search the corpus for sentences containing all arguments of a test instance (explicitly or via coreference). We randomly sampled from this set, choosing at most 10 sentences for each test instance, to form a collection of at most 200 sentences likely to be texts expressing the desired relation. These sentences were then manually annotated in the same manner as the precision annotation. Sentences that did not correctly convey the relation instance were removed, and the remaining set of sentences formed a recall set. We consider a recall set instance to be found by a system if the system finds a relation of the correct type in the sentence. We intentionally chose to sample 10 sentences from each test example, rather than sampling from the set of all sentences found. This prevents one or two very commonly expressed instances from dominating the recall set. As a result, the recall test set is biased away from "true" recall, because it places a higher weight on the "long tail" of instances. However, this gives a more accurate indication of the system's ability to find novel instances of a relation.

[8] An instance like *hasChild(his father, he)* would be useful neither during training nor (without coreference) at runtime.
[9] An average of 12,583 matches versus 2,256 matches. If multiple mentions expressing an argument occur in one sentence, each match is counted, inflating the difference.
[10] Despite being selected to be rich in the 18 ACE relation subtypes, the 10 most frequent subtypes account for over 90% of the relations with the 4 most frequent accounting for 62%; the 5 least frequent relation subtypes occur less than 50 times.

5 Empirical Results

Table 1 gives results for precision, recall, and F for +Coref (+) and –Coref (-). In all cases removing coreference causes a drop in recall, ranging from only 33%(*hasBirthPlace*) to over 90% (*GPEEmploys*). The median drop is 68%.

	P+	P-	R+	R-	R*	F+	F-
attendSchool (1)	83	**97**	**49**	16	27	**62**	27
GPEEmploy(2)	91	**96**	**29**	3	3	**44**	5
GPELeader (3)	87	**99**	**48**	28	30	**62**	43
hasBirthPlace (4)	87	**97**	**57**	37	53	**69**	53
hasChild (5)	**70**	60	**37**	17	11	**48**	27
hasSibling (6)	**73**	69	**67**	17	17	**70**	28
hasSpouse (7)	61	**96**	**72**	22	31	**68**	36
ORGEmploys(8)	**92**	82	**22**	4	7	**35**	7
ORGLeader (9)	88	**97**	**73**	32	42	**80**	48
hasBirthDate (10)	**90**	85	**45**	13	32	**60**	23

Table 1: Precision, Recall, and F scores

5.1 Recall

There are two potential sources of –Coref's lower recall. For some relation instances, the text will contain only non-named instances, and as a result -Coref will be unable to find the instance. -Coref is also at a disadvantage while learning, since it has access to fewer texts during bootstrapping. Figure 3[11] presents the fraction of instances in the recall test set for which both argument names appear in the sentence. Even with perfect patterns, -Coref has no opportunity to find roughly 25% of the relation texts because at least one argument is not expressed as a name.

To further understand -Coref's lower performance, we created a third system, *Coref, which used coreference at runtime but not during training.[12] In a few cases, such as *hasBirthPlace*, *Coref is able to almost match the recall of the system that used coreference during learning (+Coref), but on average the lack of coreference at runtime accounts for only about 25% of the difference, with the rest accounted for by differences in the pattern sets learned.

Figure 4 shows the distribution of argument mention types for +Coref on the recall set. Comparing this to Figure 3, we see that +Coref uses name-name pairs far less often than it could (less

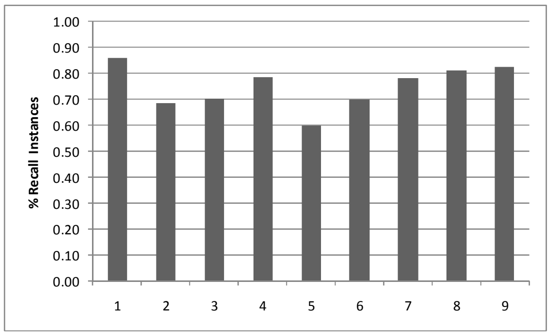

Figure 3: Fraction of recall instances with name mentions present in the sentence for both arguments.

than 50% of the time overall). Instead, even when two names are present in a sentence that entails the relation, +Coref chooses to find the relation in name-descriptor and name-pronoun contexts which are often more locally related in the sentences.

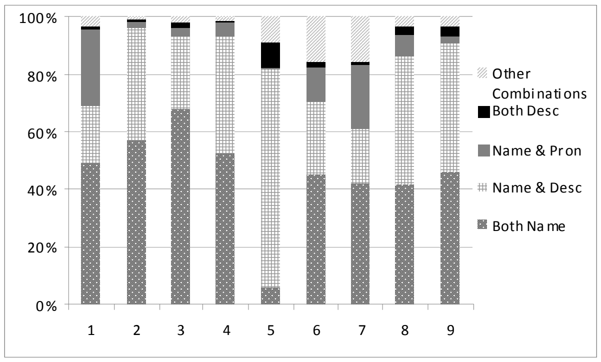

Figure 4: Distribution of argument mention types for +Coref matches on the recall set

For the two cases with the largest drops in recall, *ORGEmploys* and *GPEEmploys*, +Coref and –Coref have very different trajectories during training. For example, in the first iteration, –Coref learns patterns involving *director*, *president*, and *head* for *ORGEmploys*, while +Coref learns patterns involving *joined* and *hired*. We speculate that –Coref may become stuck because the most frequent name-name constructions, e.g. *ORG/GPE title PERSON (e.g. Brazilian President Lula da Silva)*, are typically used to introduce top officials. For such cases, even without co-reference, system specific effort and tuning could potentially have improved –Coref's ability to learn the relations.

5.2 Precision

Results on precision are mixed. While for 4 of the relations +Coref is higher, for the 6 others the addition of coreference reduces precision. The average precisions for +Coref and –Coref are 82.2 and 87.8, and the F-score of +Coref exceeded that

[11] Figures 3 & 4 do not include *hasBirthDate:* There is only 1 potential named argument for this relation, the other is a date.

[12] *Coref was added after reading paper reviews, so there was not time to do annotation for a precision evaluation for it.

of –Coref for all relations. Thus while +Coref pays a price in precision for its improved recall, in many applications it may be a worthwhile tradeoff.

Though one might expect that errors in coreference would reduce precision of +Coref, such errors may be balanced by the need to use longer patterns in –Coref. These patterns often include error-prone wildcards which lead to a drop in precision. Patterns with multiple wildcards were also more likely to be removed as unreliable in manual pattern pruning, which may have harmed the recall of –Coref, while improving its precision.

5.3 Further Analysis

Our analysis thus far has focused on micro-reading which requires a system find all mentions of an instance relation – i,e, in our evaluation *OrgLeader(Apple, Steve Jobs)* might occur in as many as 20 different contexts. While –Coref performs poorly at micro-reading, it could still be effective for macro-reading, i.e. finding at least one instance of the relation *OrgLeader(Apple, Steve Jobs)*. As a rough measure of this, we also evaluated recall by counting the number of test instances for which at least one answer was found by the two systems. With this method, +Coref's recall is still higher for all but one relation type, although the gap between the systems narrows somewhat.

	+Coref	-Coref	#Test Instances
ORGEmploys	**8**	2	20
GPEEmploys	**12**	3	19
hasSibling	**11**	4	19
hasBirthDate	**12**	5	17
hasSpouse	**15**	9	20
ORGLeader	**14**	9	19
attendedSchool	**17**	12	20
hasBirthPlace	**19**	15	20
GPELeader	**15**	13	19
hasChild	**6**	**6**	19

Table 2: Number of test seeds where at least one instance is found in the evaluation.

In addition to our recall evaluation, we measured the number of sentences containing relation instances found by each of the systems when applied to 5,000 documents (see Table 3). For almost all relations, +Coref matches many more sentences, including finding more sentences for those relations for which it has higher precision.

6 Conclusion

	Prec		Number of Sentences		
Relation	P+	P-	+Cnt	-Cnt	*Cnt
attendedSchool	83	**97**	541	212	**544**
hasChild	91	**96**	**661**	68	106
hasSpouse	87	**99**	**1262**	157	282
hasSibling	87	**97**	**313**	72	272
GPEEmploys	**70**	60	**1208**	308	313
GPELeader	**73**	69	**1018**	629	644
ORGEmploys	61	**96**	**1698**	142	209
ORGLeader	**92**	82	**1095**	207	286
hasBirthDate	88	**97**	**231**	131	182
hasBirthPlace	**90**	85	**836**	388	558

Table 3: Number of sentences in which each system found relation instances

Our experiments suggest that in contexts where recall is important incorporating coreference into a relation extraction system may provide significant gains. Despite being noisy, coreference information improved F-scores for all relations in our test, more than doubling the F-score for 5 of the 10.

Why is the high error rate of coreference not very harmful to +Coref? We speculate that there are two reasons. First, during training, not all coreference is treated equally. If the only evidence we have for a proposed instance depends on low confidence coreference links, it is very unlikely to be added to our instance set for use in future iterations. Second, for both training and runtime, many of the coreference links relevant for extracting the relation set examined here are fairly reliable, such as *wh*-words in relative clauses.

There is room for more investigation of the question, however. It is also unclear if the same result would hold for a very different set of relations, especially those which are more event-like than relation-like.

Acknowledgments

This work was supported, in part, by DARPA under AFRL Contract FA8750-09-C-179. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We would like to thank our reviewers for their helpful comments and Martha Friedman, Michael Heller, Elizabeth Roman, and Lorna Sigourney for doing our evaluation annotation.

References

E. Agichtein and L. Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *Proceedings of the ACM Conference on Digital Libraries*, pp. 85-94.

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

A. Baron and M. Freedman. 2008. Who is Who and What is What: Experiments in Cross Document Co-Reference. In *Empirical Methods in Natural Language Processing*.

A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the 1998 Conference on Computational Learning Theory*.

E. Boschee, V. Punyakanok, R. Weischedel. 2008. An Exploratory Study Towards 'Machines that Learn to Read'. *Proceedings of AAAI BICA Fall Symposium*.

J. Chen, D. Ji, C. Tan and Z. Niu. 2006. Relation extraction using label propagation based semi-supervised learning. *COLING-ACL 2006*: 129-136.

T. Mitchell, J. Betteridge, A. Carlson, E. Hruschka, and R. Wang. 2009. Populating the Semantic Web by Macro-Reading Internet Text. Invited paper, *Proceedings of the 8th International Semantic Web Conference (ISWC 2009).*

National Institute of Standards and Technology. 2007. NIST 2007 Automatic Content Extraction Evaluation Official Results. http://www.itl.nist.gov/iad/mig/tests/ace/2007/doc/ace07_eval_official_results_20070402.html

P. Pantel and M. Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of Conference on Computational Linguistics / Association for Computational Linguistics (COLING/ACL-06)*. pp. 113-120. Sydney, Australia.

L. Ramshaw, E. Boschee, S. Bratus, S. Miller, R. Stone, R. Weischedel, A. Zamanian. 2001. Experiments in multi-modal automatic content extraction, In *Proceedings of Human Language Technology Conference*.

L. Ramshaw, E. Boschee, M. Freedman, J. MacBride, R. Weischedel, A. Zamanian. 2011. SERIF Language Processing – Efficient Trainable Language Understanding. In *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*. Springer.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 41–47, Philadelphia, PA.

E. Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044-1049.

G. Zhou, J. Li, L. Qian, Q. Zhu. 2008. Semi-Supervised Learning for Relation Extraction. *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

Z. Kozareva and E. Hovy. Not All Seeds Are Equal: Measuring the Quality of Text Mining Seeds. 2010. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* pp. 618-626.