Empirical Studies in Learning to Read

Marjorie Freedman, Edward Loper, Elizabeth Boschee, Ralph Weischedel
BBN Raytheon Technologies
10 Moulton St
Cambridge, MA 02139
{mfreedma, eloper, eboschee, weischedel}@bbn.com

Abstract

In this paper, we present empirical results on the challenge of *learning to read*. That is, given a handful of examples of the concepts and relations in an ontology and a large corpus, the system should learn to map from text to the concepts/relations of the ontology. In this paper, we report contrastive experiments on the recall, precision, and F-measure (F) of the mapping in the following conditions: (1) employing word-based patterns, employing semantic structure, and combining the two; and (2) fully automatic learning versus allowing minimal questions of a human informant.

1 Introduction

This paper reports empirical results with an algorithm that "learns to read" text and map that text into concepts and relations in an ontology specified by the user. Our approach uses unsupervised and semi-supervised algorithms to harness the *diversity* and *redundancy* of the ways concepts and relations are expressed in document collections. Diversity can be used to automatically generate patterns and paraphrases for new concepts and relations to boost recall. Redundancy can be exploited to automatically check and improve the accuracy of those patterns, allowing for system learning without human supervision.

For example, the system learns how to recognize a new relation (e.g. *invent*), starting from 5-20 instances (e.g. *Thomas Edison* + *the light bulb*). The system iteratively searches a collection of documents to find sentences where those instances are expressed (e.g. "*Thomas Edison's patent for the light bulb*"), induces patterns over textual features found in those instances (e.g. *patent(possessive:A, for:B)*), and repeats the cycle by applying the generated patterns to find additional instances followed by inducing more patterns from those instances. Unsupervised measures of redundancy and coverage are used to estimate the reliability of the induced patterns and learned instances; only the most reliable are added, which minimizes the amount of noise introduced at each step.

There have been two approaches to evaluation of mapping text to concepts and relations: Automatic Content Extraction (ACE)[1] and Knowledge Base Population (KBP)[2]. In ACE, complete manual annotation for a small corpus (~25k words) was possible; thus, both recall and precision could be measured across every instance in the test set. This evaluation can be termed *micro reading* in that it evaluates every concept/relation mention in the corpus. In ACE, learning algorithms had roughly 300k words of training data.

By contrast, in KBP, the corpus of documents in the test set was too large for a complete answer key. Rather than a complete answer key, relations were extracted for a list of entities; system output was pooled and judged manually. This type of reading has been termed *macro reading*[3], since finding any instance of the relation in the 1.3M document corpus is measured success, rather than finding every instance. Only 118 queries were provided, though several hundred were created and distributed by participants.

In the study in this paper, recall, precision, and F are measured for 11 relations under the following contrastive conditions

[1] http://www.nist.gov/speech/tests/ace/
[2] http://apl.jhu.edu/~paulmac/kbp.html
[3] See http://rtw.ml.cmu.edu/papers/mitchell-iswc09.pdf

Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading, pages 61–69,
Los Angeles, California, June 2010.

1. Patterns based on words vs. predicate-argument structure vs. combining both.
2. Fully automatic vs. a few periodic responses by humans to specific queries.

Though many prior studies have focused on precision, e.g., to find any text justification to answer a question, we focus equally on recall and report recall performance as well as precision. This addresses the challenge of finding information on rarely mentioned entities (no matter how challenging the expression). We believe the effect will be improved technology overall. We evaluate our system in a micro-reading context on 11 relations. In a fully automatic configuration, the system achieves an F of .48 (Recall=.37, Precision=.68). With limited human intervention, F rises to .58 (Recall=.49, Precision=.70). We see that patterns based on predicate-argument structure (text graphs) outperform patterns based on surface strings with respect to both precision and recall.

Section 2 describes our approach; section 3, some challenges; section 4, the implementation; section 5, evaluation; section 6, empirical results on extraction type; section 7, the effect of periodic, limited human feedback; section 8, related work; and section 9, lessons learned and conclusions.

2 Approach

Our approach for learning patterns that can be used to detect relations is depicted in Figure 1. Initially, a few instances of the relation tuples are provided, along with a massive corpus, e.g., the web or the gigaword corpus from the Linguistic Data Consortium (LDC). The diagram shows three inventor-invention pairs, beginning with *Thomas Edison…light bulb*. From these, we find candidate sentences in the massive corpus, e.g., *Thomas Edison invented the light bulb*. Features extracted from the sentences retrieved, for example features of the text-graph (the predicate-argument structure connecting the two arguments), provide a training instance for pattern induction. The induced patterns are added to the collection (database) of patterns. Running the extended pattern collection over the corpus finds new, previously unseen relation tuples. From these new tuples, additional sentences which express those tuples can be retrieved, and the cycle of learning can continue.

There is an analogous cycle of learning concepts from instances and the large corpus; the experiments in this paper do not report on that parallel learning cycle.

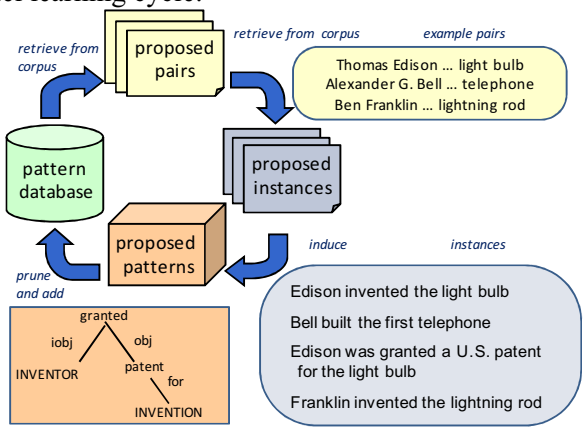


Figure 1: Approach to Learning Relations

At the i^{th} iteration, the steps are

1. Given the set of hypothesized instances of the relation (triples HT_i), find instances of such triples in the corpus. (On the first iteration, "hypothesized" triples are manually-generated seed examples.)
2. Induce possible patterns. For each proposed pattern P:
   a. Apply pattern P to the corpus to generate a set of triples T_P
   b. Estimate precision as the confidence-weighted average of the scores of the triples in T_P. Reduce precision score by the percentage of triples in T_P that violate user-specified relation constraints (e.g. arity constraints described in 4.3)
   c. Estimate recall as the confidence-weighted percentage of triples in HT_i found by the pattern
3. Identify a set of high-confidence patterns HP_i using cutoffs automatically derived from rank-based curves for precision, recall, and F-measure (α=0.7)
4. Apply high-confidence patterns to a Web-scale corpus to hypothesize new triples. For each proposed triple T
   a. Estimate *score*(*T*) as the expected probability that T is correct, calculated by combining the respective precision and recall scores of all of the patterns that did or did not return it (using the Naïve Bayes assumption that all patterns are independent)
   b. Estimate *confidence*(*T*) as the percentage of patterns in HP_i by which T was found
5. Identify a set of high-confidence triples HT_{i+1}

using cutoffs automatically derived from rank-based curves; use these triples to begin the next iteration

3 Challenges

The iterative cycle of learning we describe above has most frequently been applied for *macro-reading* tasks. However, here we are interested in measuring performance for *micro-reading.* There are several reasons for wanting to measure performance in a micro-reading task:

- For information that is rare (e.g. relations about infrequently mentioned entities), a micro-reading paradigm may more accurately predict results.
- For some tasks or domains micro-reading may be all that we can do-- the actual corpus of interest may not be macro-scaled.
- Macro-reading frequently utilizes statistics of extraction targets from the whole corpus to improve its precision. Therefore, improving micro-reading could improve the precision of macro-reading.
- Because we are measuring performance in a micro-reading context, recall at the instance level is as important as precision. Consequently, our learning system must learn to predict patterns that incorporate nominal and pronominal mentions.
- Furthermore, while the approach we describe makes use of corpus-wide statistics during the learning process, during pattern application we limit ourselves to only information from within a single document (and in practice primarily within a single sentence). Our evaluation measures performance at the instance-level[4].

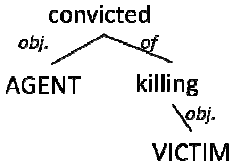
4 Implementation

4.1 Pattern Types

Boschee et al.(2008) describes two types of patterns: patterns that rely on surface strings and patterns that rely only on two types of syntactic-structure. We diverge from that early work by allowing more expressive surface-string patterns: our surface-string patterns can include wild-cards (allowing the system to make matches which require omitting words). For example for *kill(agent, victim)*, the system learns the pattern *<AGENT> <...> assassinated <VICTIM>*, which correctly matches *Booth, with hopes of a resurgent Confederacy in mind, cruelly assassinated Lincoln*.

We also diverge from the earlier work by only making use of patterns based on the normalized predicate-argument structure and not dependency parses. The normalized predicate-argument structures (text-graphs) are built by performing a set of rule-based transformations on the syntactic parse of a sentence. These transformations include finding the logical subject and object for each verb, resolving some traces, identifying temporal arguments, and attaching other verb arguments with lexicalized roles (e.g. '*of*' in Figure 2). The resulting graphs allow both noun and verb predicates.

Manually created patterns using this structure have been successfully used for event detection and template-based question answering. The text graph structures have also served as useful features in supervised, discriminative models for relation and event extraction. While the experiments described here do not include dependency tree paths, we do allow arbitrarily large text graph patterns.



Text Graph Pattern

4.2 Co-Reference

Non-named mentions are essential for allowing high instance-level recall. In certain cases, a relation is most clearly and frequently expressed with pronouns and descriptions (e.g *her father* for the relation *child).*[5] Because non-named instances appear in different constructions than named instances, we need to learn patterns that will appear in non-named contexts. Thus, co-reference information is used during pattern induction to extract patterns from sentences where the hypothesized triples are not explicitly mentioned. In particular, any mention that is co-referent with the desired entity can be used to induce a pattern. Co-reference for 7 types of entities is produced by SERIF, a

[4]Our evaluation measures only performance in extracting the relation: that is if the text of sentence implies to an annotator that the relation is present, then the annotator has been instructed to mark the sentence as correct (regardless of whether or not outside knowledge contradicts this fact).

[5] The structure of our noun-predicates allows us to learn lexicalized patterns in cases like this. For *her father* we would induce the pattern *n:father:<ref>PARENT, <pos>CHILD.*

state of the art information extraction engine. A manually determined confidence threshold is used to discard mentions where co-reference certainty is too low.

During pattern matching, co-reference is used to find the "best string" for each element of the matched triple. In particular, pronouns and descriptors are replaced by their referents; and abbreviated names are replaced by full names. If any pronoun cannot be resolved to a description or a name, or if the co-reference threshold falls below a manually determined threshold, then the match is discarded.

Pattern scoring requires that we compare instances of triples across the whole corpus. If these instances were compared purely on the basis of strings, in many cases the same entity would appear as distinct *(*e.g. *US, United States)*. This would interfere with the arity constraints described below. To alleviate this challenge, we use a database of name strings that have been shown to be equivalent with a combination of edit-distance and extraction statistics (Baron & Freedman, 2008). Thus, for triple(t_P) and hypothesized triples (HT_i), if $t_P \notin HT_i$, but can be mapped via the equivalent names database to some triple $t_P' \in HT_i$, then its score and confidence are adjusted towards that of t_P', weighted by the confidence of the equivalence.

4.3 Relation Set and Constraints

We ran experiments using 11 relation types. The relation types were selected as a subset of the relations that have been proposed for DARPA's machine reading program. In addition to seed examples, we provided the learning system with three types of constraints for each relation:

Symmetry: For relations where R(X,Y) = R(Y,X), the learning (and scoring process), normalizes instances of the relation so that R(X,Y) and R(Y,X) are equivalent.

Arity: For each argument of the relation, provide an expected maximum number of instances per unique instance of the other argument. These numbers are intentionally higher than the expected true value to account for co-reference mistakes. Patterns that violate the arity constraint (e.g *v:accompanied(<obj>=<X>, <sub>=<Y>* as a pattern for *spouse)* are penalized. This is one way of providing negative feedback during the unsupervised training process.

Argument Type: For each argument, specify the expected class of entities for this argument. Entity types are one of the 7 ACE types (Person, Organization, Geo-political entity, Location, Facility, Weapon, Vehicle) or Date. Currently the system only allows instance proposals when the types are correct. Potentially, the system could use pattern matches that violate type constraints as an additional type of negative example. Any implementation would need to account for the fact that in some cases, potentially too general patterns are quite effective when the type constraints are met. For example, for the relation *employed(PERSON, ORGANIZATION),* <ORG>*'s* <PER> is a fairly precise pattern, despite clearly being overly general without the type constraints.

In our relation set, only two relations (*sibling* and *spouse*) are symmetric. Table 1 below includes the other constraints. ACE types/dates are in columns labeled with the first letter of the name of the type (A is arity). We have only included those types that are an argument for some relation.

Relation	Arg1					Arg2						
	P	O	G	L	A	P	O	G	F	L	D	A
attackedOn	■	■	■	■	10						■	--
attendSchool	■				--		■					5
employed	■				--		■	■		■		5
birthDate	■				--						■	5
birthplace	■				--			■		■		5
child	■				5	■						10
kill	■				--	■						--
killedInLocation	■				--			■	■	■		5
sibling	■				10	■						10
spouse	■				5	■						5
topLeader	■				5		■	■		■		5

Table 1: Argument types of the test relations

4.4 Corpus and Seed Examples

While many other experiments using this approach have used web-scale corpora, we chose to include Wikipedia articles as well as Gigaword-3 to provide additional instances of information (e.g. *birthDate* and *sibling*) that is uncommon in news.

For each relation type, 20 seed-examples were selected randomly from the corpus by using a combination of keyword search and an ACE extraction system to identify passages that were likely to contain the relations of interest. As such, each seed example was guaranteed to occur at least once in a context that indicated the relation was present.

5 Evaluation Framework

To evaluate system performance, we ran two sepa-

rate annotation-based evaluations, the first measured precision, and the second measured recall.

To measure overall precision, we ran each system's patterns over the web-scale corpora, and randomly sampled 200 of the instances it found[6]. These instances were then manually assessed as to whether they conveyed the intended relation or not. The system precision is simply the percentage of instances that were judged to be correct.

To measure recall, we began by randomly selecting 20 test-examples from the corpus, using the same process that we used to select the training seeds (but guaranteed to be distinct from the training seeds). We then searched the web-scale corpus for sentences that might possibly link these test examples together (whether directly or via coreference). We randomly sampled this set of sentences, choosing 10 sentences for each test-example, to form a collection of 200 sentences which were likely to convey the desired relation. These sentences were then manually annotated to indicate which sentences actually convey the desired relation; this set of sentences forms the *recall test set*. Once a recall set had been created for each relation, a system's recall could be evaluated by running that system's patterns over the documents in the recall set, and checking what percentage of the recall test sentences it correctly identified.

We intentionally chose to sample 10 sentences from each test example, rather than sampling from the set of all sentences found for any of the test examples, in order to prevent one or two very common instances from dominating the recall set. As a result, the recall test set is somewhat biased away from "true" recall, since it places a higher weight on the "long tail" of instances. However, we believe that this gives a more accurate indication of the system's ability to find novel instances of a relation (as opposed to novel ways of talking about known instances).

6 Effect of Pattern Type

As described in 4.1, our system is capable of learning two classes of patterns: surface-strings and text-graphs. We measured our system's performance on each of the relation types after twenty iterations. In each iteration, the system can learn multiple patterns of either type. There is currently no penalty for learning overlapping pattern types. For example, in the first iteration for the relation *killed*(), the system learns both the surface-string pattern *<AGENT> killed <VICTIM>* and the text-graph pattern: *v:killed(<sub>=<AGENT>, <obj>=<VICTIM>)*. During decoding, if multiple patterns match the same relation instance, the system accepts the relation instance, but does not make use of the additional information that there were multiple supporting patterns.

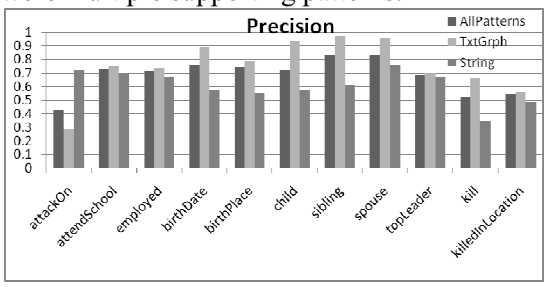


Figure 3: Precision of Pattern Types by Relation

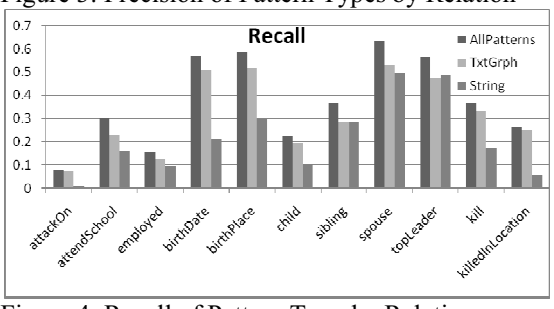


Figure 4: Recall of Pattern Type by Relation

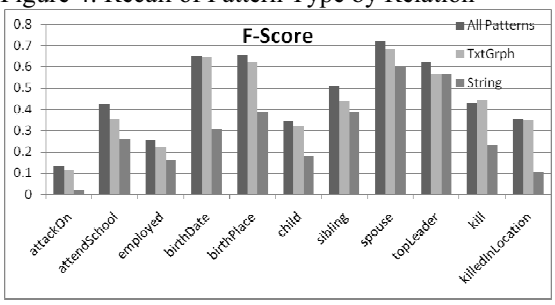


Figure 5: F-Score of Pattern Type by Relation

Figure 3, Figure 4, and Figure 5 plot precision, recall, and F-score for each of the 11 relations showing performance of all patterns vs. only text-graph patterns vs. only surface-string patterns.

- For most relations, the text-graph patterns provide both higher precision and higher recall than the surface-string patterns. The lower precision of the text-graph patterns for *attackOn* is the result of the system learning a number of overly general patterns that correlate with attacks, but do not themselves indicate the pres-

[6] While the system provides estimated precision for each pattern, we do not evaluate over the n-best matches. All patterns with estimated confidence above 50% are treated equally. We sample from the set of matches produced by these patterns.

ence of an attack. For instance, the system learns patterns with predicates *said* and *arrive.* Certainly, governments often make statements on the date of an attack and troops arrive in a location before attacking, but both patterns will produce a large number of spurious instances.

- While text-graph patterns typically have higher precision than the combined pattern set, surface-string patterns provide enough improvement in recall that typically the all-pattern F-score is higher than the text-graph F-score.

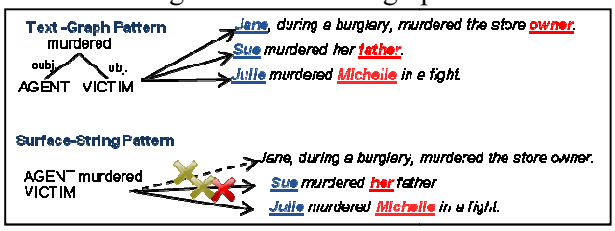


Figure 6: Text-Graph and Surface-String Patterns

A partial explanation for the higher recall and precision of the text-graph patterns is illustrated in Figure 6 which presents a simple surface-string pattern and a simple text-graph pattern that appear to represent the same information. On the right of the figure are three sentences. The text-graph pattern correctly identifies the *agent* and *victim* in each sentence. However, the surface-string pattern, misses the *killed*() relation in the first sentence and misidentifies the *victim* in the second sentence. The false-alarm in the second sentence would have been avoided by a system that restricted itself to matching named instances, but as described above in section 4.2, for the micro-reading task described here, detecting relations with pronouns is critical.

While we allowed the creation of text-graph patterns with arbitrarily long paths between the arguments, in practice, the system rarely learned such patterns. For the relation *killed(Agent, Victim)*, we learned 8 patterns that have more than one predicate (compared to 22 that only have a single predicate). For the relation *killedInLocation(Victim, Location)*, the system learned 28 patterns with more than 1 predicate (compared with 20 containing only 1 predicate). In both cases, the precision of the longer patterns was higher, but their recall was significantly lower. In the case of *killedInLocation,* none of the longer path patterns matched any of the examples in our recall set.

One strength of text-graph patterns is allowing for intelligent omission of overly specific text, for example, ignoring '*during a buglary'* in Figure 6. Surface string patterns can include wild-cards, but for surface string patterns, the omission is not syntactically defined. Approximately 30% of surface string patterns included one wild-card. An additional 17% included two. Figure 7 presents averaged precision and recall for text-graph and surface-string patterns. The final three columns break the surface-string patterns down by the number of wild-cards. It appears that with one, the patterns remain reasonably precise, but the addition of a second wild-card drops precision by more than 50%. The presence of wild-card patterns improves recall, but surface-string patterns do not reach the level of recall of text-graph patterns.

	Text Graph	Surface String			
		All	No-*	1-*	2-*
Precision	0.75	0.61	0.72	0.69	0.30
Recall	0.32	0.22	0.16	0.10	0.09

Figure 7: Performance by Number of WildCards (*)

7 Effect of Human Review

In addition to allowing the system to self-train in a completely unsupervised manner, we ran a parallel set of experiments where the system was given limited human guidance. At the end of iterations 1, 5, 10, and 20, a person provided under 10 minutes of feedback (on average 5 minutes). The person was presented with five patterns, and five sample matched instances for each pattern. The person was able to provide two types of feedback:

- The pattern is correct/incorrect (e.g. <EMPLOYEE> *said* <ORGANIZATION> is an incorrect pattern for employ(X,Y))
- The matched instances are correct/incorrect (e.g. '*Bob received a diploma from MIT'* is a correct instance, even if the pattern that produced it is debatable (e.g. *v:<received> subj:PERSON, from:ORGANIZATION*). A correct instance can also produce a new known-to-be correct seed.

Pattern judgments are stored in the database and incorporated as absolute truth. Instance judgments provide useful input into pattern scoring. Patterns were selected for annotation using a score that combines their estimated f-measure; their frequency; and their dissimilarity to patterns that were previously chosen for annotation. The matched instances for each pattern are randomly sampled, to ensure that the resulting annotation can be used to derive an unbiased precision estimate.

Figure 8, Figure 9, and Figure 10 plot precision, recall, and F-score at iterations 5 and 20 for the system running in a fully unsupervised manner and one allowing human intervention.

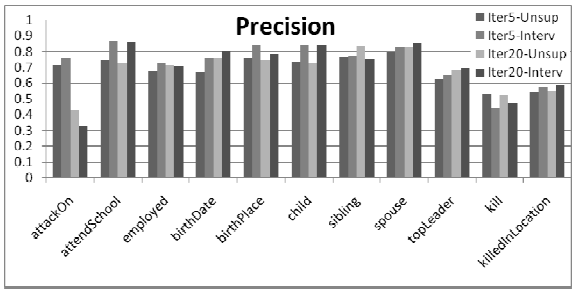


Figure 8: Precision at Iterations 5 and 20 for the Unsupervised System and the System with Intervention

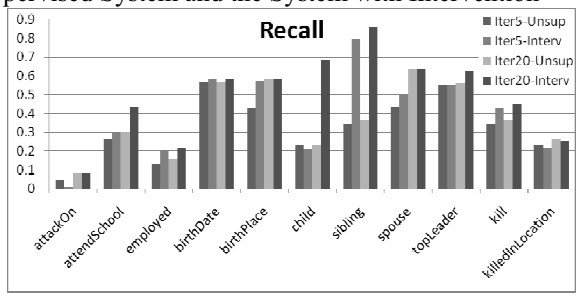


Figure 9: Recall at Iterations 5 and 20 for the Unsupervised System and the System with Intervention

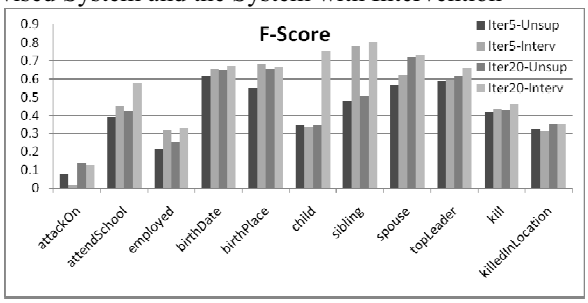


Figure 10: F-Score at Iterations 5 and 20 for the Unsupervised System and the System with Intervention

For two relations: *child* and *sibling,* recall improved dramatically with human intervention. By inspecting the patterns produced by the system, we see that in case of *sibling* without intervention, the system only learned the relation '*brother'* and not the relation '*sister.'* The limited feedback from a person was enough to allow the system to learn patterns for *sister* as well, causing the significantly improved recall. We see smaller, but frequently significant improvements in recall in a number of other relations. Interestingly, for different relations, the recall improvements are seen at different iterations. For *sibling,* the jump in recall appears within the first five iterations. Contrastingly, for *attendSchool*, there is a minor improvement in recall after iteration 5, but a much larger improvement after iteration 20. For *child,* there is actually a small decrease in recall after 5 iterations, but after 20 iterations, the system has dramatically improved.

The effect on precision is similarly varied. For 9 of the 11, human intervention improves precision; but the improvement is never as dramatic as the improvement in recall. For precision, the strongest improvements in performance appear in the early iterations. It is unclear whether this merely reflects that bootstrapping is likely to become less precise over time (as it learns more patterns), or if early feedback is truly better for improving precision.

In the case of *attackOn,* even with human intervention, after iteration 10, the system begins to learn very general patterns of the type described in the previous section (e.g. *<said in:LOCATION on:DATE>* as a pattern indicating an attack. These patterns may be correlated with experiencing an attack but are not themselves evidence of an attack. Because the overly general patterns do in fact correlate with the presence of an attack, the positive examples provided by human intervention may in fact produce more such patterns.

There is an interaction between improved precision and improved recall. If a system is very imprecise at iteration *n*, the additional instances that it proposes may not reflect the relation and be so different from each other that the system becomes unable to produce good patterns that improve recall. Conversely, if recall at iteration *n* does not produce a sufficiently diverse set of instances, it will be difficult for the system to generate instances that are used to estimate pattern precision.

8 Related Work

Much research has been done on concept and relation detection using large amounts of supervised training. This is the typical approach in programs like Automatic Content Extraction (ACE), which evaluates system performance in detecting a fixed set of concepts and relations in text. In ACE, all participating researchers are given access to a substantial amount of supervised training, e.g., 250k words of annotated data. 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 the

manually annotated examples, which are expensive and time-consuming to create.

Co-training circumvents this weakness by playing off two sufficiently different views of a data set to leverage large quantities of unlabeled data (along with a few examples of labeled data), in order to improve the performance of a learning algorithm (Mitchell and Blum, 1998). Co-training will offer our approach to simultaneously learn the patterns of expressing a relation and its arguments.

Other researchers have also previously explored automatic pattern generation from unsupervised text, classically in (Riloff & Jones 1999). Ravichandran and Hovy (2002) reported experimental results for automatically generating surface patterns for relation identification; others have explored similar approaches (e.g. Agichtein & Gravano 2000 or Pantel & Pennacchiotti, 2006). More recently (Mitchell et al., 2009) has shown that for macro-reading, precision and recall can be improved by learning a large set of interconnected relations and concepts simultaneously.

We depart from this work by learning patterns that use the structural features of text-graph patterns and our particular approach to pattern and pair scoring and selection.

Most approaches to automatic pattern generation have focused on precision, e.g., Ravichandran and Hovy report results in the Text Retrieval Conference (TREC) Question Answering track, where extracting one instance of a relation can be sufficient, rather than detecting all instances. This study has also emphasized recall. Information about an entity may only be mentioned once, especially for rarely mentioned entities. A primary focus on precision allows one to ignore many instances that require co-reference or long-distance dependencies; one primary goal of our work is to measure system performance in exactly those areas.

9 Conclusion

We have shown that bootstrapping approaches can be successfully applied to micro-reading tasks. Most prior work with this approach has focused on macro-reading, and thus emphasized precision. Clearly, the task becomes much more challenging when the system must detect every instance. Despite the challenge, with very limited human intervention, we achieved F-scores of >.65 on 6 of the 11 relations (average F on the relation set was .58).

We have also replicated an earlier preliminary result (Boschee, 2008) showing that for a micro-reading task, patterns that utilize semantic/syntactic information outperform patterns that make use of only surface strings. Our result covers a larger inventory of relation types and attempts to provide a more precise measure of recall than the earlier preliminary study.

Analysis of our system's output provides insights into challenges that such a system may face.

One challenge for bootstrapping systems is that it is easy for the system to learn just a subset of relations**.** We observed this in both *sibling* where we learned the relation *brother* and for *employed* where we only learned patterns for leaders of an organization. For *sibling* human intervention allowed us to correct for this mistake. However for *employed* even with human intervention, our recall remains low. The difference between these two relations may be that for *sibling* there are only two sub-relations to learn, while there is a rich hierarchy of potential sub-relations under the general relation *employed.* The challenge is quite possibly exacerbated by the fact that the distribution of employment relations in the news is heavily biased towards top officials, but our recall test set intentionally does not reflect this skew.

Another challenge for this approach is continuing to learn in successive iterations. As we saw in the figures in Section 7, for many relations performance at iteration 20 is not significantly greater than performance at iteration 5. Note that seeing improvements on the long tail of ways to express a relation may require a larger recall set than the test set used here. This is exemplified by the existence of the highly precise 2-predicate patterns which in some cases never fired in our recall test set.

In future, we wish to address the subset problem and the problem of stalled improvements. Both could potentially be addressed by improved internal rescoring. For example, the system scoring could try to guarantee coverage over the whole seed-set thus promoting patterns with low recall, but high value for reflecting different information. A complementary set of improvements could explore improved uses of human intervention.

Acknowledgments

This work was supported, in part, by BBN under AFRL Contract FA8750-09-C-179.

References

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

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

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

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

M. Collins and Y Singer. Unsupervised Models for Named Entity Classification. EMNLP/VLC. (1999).

M Mintz, S Bills, R Snow, and D Jurafsky.. Distant supervision for relation extraction without labeled data. Proceedings of ACL-IJCNLP 200. 2009..

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

P. Pantel and M. Pennacchiotti. 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, 2006.

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

D. Ravichandran and E. Hovy. 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, 2002.

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

E. Rilof and Jones, R "Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping", Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) , 1999, pp. 474-479. 1999.

R Snow, D Jurafsky, and A Y. Ng.. Learning syntactic patterns for automatic hypernym discovery . Proceedings of NIPS 17. 2005.