Extreme Extraction -- Machine Reading in a Week

Marjorie Freedman, Lance Ramshaw, Elizabeth Boschee, Ryan Gabbard, Gary Kratkiewicz, Nicolas Ward, Ralph Weischedel
Raytheon BBN Technologies
10 Moulton St.
Cambridge, MA 02138
mfreedma,lramshaw,eboschee,rgabbard,kratkiewicz,
nward,weischedel@bbn.com

Abstract

We report on empirical results in *extreme extraction*. It is extreme in that (1) from receipt of the ontology specifying the target concepts and relations, development is limited to one week and that (2) relatively little training data is assumed. We are able to surpass human recall and achieve an F1 of 0.51 on a question-answering task with less than 50 hours of effort using a hybrid approach that mixes active learning, bootstrapping, and limited (5 hours) manual rule writing. We compare the performance of three systems: extraction with handwritten rules, bootstrapped extraction, and a combination. We show that while the recall of the handwritten rules surpasses that of the learned system, the learned system is able to improve the overall recall and F1.

1 Introduction

Throughout the Automatic Content Extraction[1] (ACE) evaluations and the Message Understanding Conferences[2] (MUC), teams typically had a year or more from release of the target to submitting system results. One exception was MUC-6 (Grishman & Sundheim, 1996), in which scenario templates for changing positions were extracted given only one month. Our goal was to confine development to a calendar week, in fact, <50 person hours. This is significant in two ways: the less effort it takes to bring up a new domain, (1) the more broadly applicable the technology is and (2) the less effort required to run a diagnostic research experiment.

Our second goal concerned minimizing training data. Rather than approximately 250k words of entity and relation annotation as in ACE, only ~20 example pairs per relation-type were provided as training. Reducing the training requirements has the same two desirable outcomes: demonstrating that the technology can be broadly applicable and reducing the overhead for running experiments.

The system achieved recall of 0.49 and precision of 0.53 (for an F1 of 0.51) on a blind test set of 60 queries of the form $R_i(arg_1, arg_2)$, where R_i is one of the 5 new relations and exactly one of arg_1 or arg_2 is a free variable for each query.

Key to this achievement was a hybrid of:

- a variant of (Miller, et al., 2004) to learn two new classes of entities via automatically induced word classes and active learning (6 hours)
- bootstrap relation learning (Freedman et al, 2010) to learn 5 new relation classes (2.5 hours),
- handwritten patterns over predicate-argument structure (5 hours), and
- coreference (20 hours)

Our bootstrap learner is initialized with relation tuples (not annotated text) and uses LDC's Gigaword and Wikipedia as a background corpus to learn patterns for relation detection that are based on normalized predicate argument structure as well as surface strings.

These early empirical results suggest the following: (1) It is possible to specify a domain, adapt our system, and complete manual scoring, includ-

[1] http://www.nist.gov/speech/tests/ace/
[2] http://www-nlpir.nist.gov/related_projects/muc/

ing human performance, within a month. Experiments in machine reading (and in extraction) can be performed much more quickly and cheaply than ever before. (2) Through machine learning and limited human pattern writing (6 hours), we adapted a machine reading system within a week (using less than 50 person hours), achieving question answering performance with an F1 of 0.5 and with recall 11% higher (relative) to a human reader. (3) Unfortunately, machine learning, though achieving 80% precision,[3] significantly lags behind a gifted human pattern writer in recall. Thus, bootstrap learning with much higher recall at minimal sacrifice in precision is highly desirable.

2 Related Work

This effort is evaluated extrinsically via formal questions expressed as a binary relation with one free variable. This contrasts with TREC Question Answering,[4] where the questions are in natural language, and not restricted to a single binary relation. Like the "list" queries of TREC QA, the requirement is to find all answers, not just one. Though question interpretation is not required in our work, interpretation of the text corpus is.

The goal of rapid adaptation has been tested in other contexts. In 2003, a series of experiments in adapting to a new language in less than month tested system performance on Cebuano and Hindi. The primary goal was to adapt to a new language, rather than a new domain. The extraction participants focused on named-entity recognition, not relation extraction (May, et al, 2003; Sekine & Grishman, 2003; Li & McCallum, 2003; Maynard et al, 2003). The scenario templates of MUC-6 (Grishman & Sundheim, 1996) are more similar to our relation extraction task, although the domain is quite different. Our experiment allowed for 1 week of development time, while MUC-6 allowed a month. The core entities in the MUC-6 task (people and organizations) had been worked on previously. In contrast all of our relations included at least one novel class. While MUC-6 systems tended to use finite-state patterns, they did not incorporate bootstrapping or patterns based on the output of a statistical parser.

For learning entity classes, we follow Miller, et al., (2004), using word clustering and active learning to train a perceptron model, but unlike that work we apply the technique not just to names but also to descriptions. An alternative approach to learning classes, applying structural patterns to bootstrap description recognition without active learning, is seen in Riloff (1996) and Kozareva et al., (2008)

Much research (e.g. Ramshaw 2001) has focused on learning relation extractors using large amounts of supervised training, as in ACE. The obvious weakness of such approaches is the resulting reliance on manually annotated examples, which are expensive and time-consuming to create.

Others have explored bootstrap relation learning 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. None use coreference to find training examples; all use surface (word) patterns. Freedman et. al (2010) report improved performance from using predicate structure for bootstrappped relation learning.

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

By contrast, our work emphasizes recall, not just precision. Our question answering task asks list-like questions that require multiple answers. We also include the results of a secondary, extraction evaluation which requires that the system identify every mention of the relations in a small set of documents. This evaluation is loosely based on the relation mention detection task in ACE.

3 Task Set-Up and Evaluation

Our effort was divided into four phases. During the first phase, a third party produced an ontology and the resources, which included: brief (~1 paragraph) guidelines for each relation and class in the ontolo-

[3] Handwritten patterns achieved 52% precision.
[4] http://trec.nist.gov/data/qamain.html

gy; ~20 examples for each relation in the ontology; 2K documents that are rich in domain relations. Table 1 lists the 5 new relations and number of examples provided for each. Arguments in italics were known by the system prior to the evaluation.

Relation	Ex.
possibleTreatment(Substance, Condition)	23
expectedDateOnMarket(Substance, *Date*)	11
responsibleForTreatment(Substance, A*gent*)	19
studiesDisease(*Agent*, Condition)	16
hasSideEffect(Substance, Condition)	27

Table 1: New Relations and Number of Examples

In phase two, we spent one week extending our extraction system for the new ontology. During the third phase, we ran our system over 10K documents to extract all instances of domain relations from those documents. In the fourth phase, our question answering system used the extracted information to answer queries.

4 Approach to Domain Specialization

Our approach to extracting domain relations integrated novel relation and class detectors into an existing extraction system, designed primarily around the ACE tasks. The existing system uses a discriminatively trained classifier to detect the entity and value types of ACE. It also produces a syntactic parse for each sentence; normalizes these parses to find logical predicate argument structure; and detects and coreferences pronominal, nominal, and name mentions for each of the 7 ACE entity types (Person, Organization, Geopolitical Entity, Location, Facility, Weapon, and Vehicle).[5]

The extraction system has three components that allow for rapid adaptation to a new domain:

- Class detectors trained using word classes derived from unsupervised clustering and sentence-selected training data.
- A bootstrap relation learner which given a few seed examples learns patterns that indicate the presence of relations.
- An expressive pattern language which allows a developer to express rules for relation extraction in a simple, but fast manner.

Component	Approach	Effort
Class Recognizer	Active Learning	6 hrs
Class Recognizer	Web-Mined List	1 hrs
Relation Recognizer	Semi-supervised Bootstrapping	8.5 hrs
Relation Recognizer	Manual Patterns	5 hrs
Coreference	Heuristics	20 hrs

Table 2: Effort and Approach for New Domain

[5] The extraction system detects relations and events in the ACE ontology, but these were not used in the current work.

4.1 Class Extraction

Each of the relations in the new domain included at least one argument that was new. While question answering requires the system to identify the classes only when they appear in a relation, knowledge of when a class is present provides important information for relation extraction. For example in our ontology, Y is a treatment for X only if Y is a substance. Thus, '*Group counseling sessions are effective treatments for depression'* does not contain an instance of *possibleTreatment*(), while '*SSRIs are effective treatments for depression*' does. The bootstrap learner allows constraints based on argument type. To use this capability, we trained the recognizer at the beginning of the week of domain adaptation and used the predicted classes during learning.

We annotated 1064 sentences (~31K words) using active learning combined with unsupervised word clusters (Miller,et al., 2004) for the following classes: *Substance-Name, Substance-Description, Condition-Name, and Condition-Description*. Generic noun-phrases like *new drugs*, *the illness*, etc were labeled as descriptors. Because of the time frame, we did not develop extensive guidelines nor measure inter-annotator agreement. Annotation took 6 hours. We supplemented our annotation with lists of substances and treatments from the web, which took 1 hour.

4.2 Coreference

Providing a name reference is generally preferable to a non-specific string (e.g. *the drugs*), but not always feasible; for instance, reports of new research may appear without a name for the drug. Our existing system's coreference algorithms operate only on mentions of ACE entity types (persons, organizations, GPEs, other locations, facilities, vehicles, and weapons). During the week of domain adaption we developed new heuristics for coreference over non-ACE types. Most of our heuristics are domain independent (e.g. linking the parts of an appositive). Our decision to annotate names and descriptions separately was driven partially by the need to select the best reference (i.e. name) for co-referent clusters. Adding coreference heuristics for the two new entity types was the single most time-consuming activity, taking 20 of the total 43 hours.

4.3 Relation Extraction

For relation extraction, we used both pattern learning and handwritten patterns. We initialized our bootstrap relation learner with the example instances provided with the domain ontology; Table 3 includes examples of the instances provided to the system as training. Our bootstrap relation learner finds instances of the relation argument pairs in text and then proposes both predicate-argument structure and word-based connections between the arguments as possible new patterns for the relation. The learner automatically prunes potential patterns using information about the number of known-to-be true and novel instances matched by a proposed pattern. By running the pattern extractor over a large corpus, the proposed patterns generate new seeds which are in turn are used to propose new patterns. For this experiment, we incorporated a small amount of supervision during the bootstrapping process (roughly 1 hour total per relation); we also performed ~30 minutes total in pruning domain patterns at the end of learning.

Relation	**Arg-1**	**Arg-2**
possTreatmnt	AZT	AIDS
studyDisease	Dr Henri Joyeux	cancer
studyDisease	Samir Khleif	cancer

Table 3: Sample Instances for Initializing Learner

We also used a small amount of human effort creating rules for detecting the relations. The pattern writer was given the guidelines, the examples, and a 2K document background corpus and spent 1 hour per relation writing rules.

The learned patterns use a subset of the full pattern language used by the pattern-writer. The language operates over surface-strings as well as predicate-argument structure. Figure 1 illustrates learned and handwritten patterns for *the possible-TreatmentRelation()*. The patterns in rectangles match surface-string patterns; the tree-like patterns match normalized predicate argument structure. The –WORD- token indicates a wild card of 1-3 words. The blue rectangles at the root of the trees in the handwritten patterns are sets of predicates that can be matched by the pattern.

5 Evaluation

Our question answering evaluation was inspired by the evaluation in DARPA's machine reading program, which requires systems to map the information in text into a formal ontology and answer questions based on that ontology. Unlike ACE, this allows evaluators to measure performance without exhaustively annotating documents, allows for balance between rare and common relations, and implicitly measures coreference without requiring explicit annotation of answer keys for coreference. However because the evaluation only measures performance on the set of queries, many relation instances will be unscored. Furthermore, the system is not rewarded for finding the same relation multiple times; finding 100 instances of *isPossibleTreatment(Penicillin, Strep Throat)* is the same as finding 1 (or 10) instances.

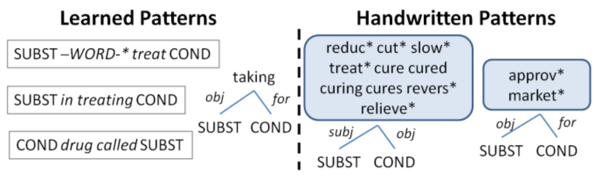

Figure 1: Sample Patterns for *possibleTreatment()*

The evaluation included only queries of the type *Find all instances for which the relation P(X, Z) is true* where one of X or Z is constant. For example, *Find possible treatments for diabetes*; or *What is expected date to market for Abilify?* There were 60 queries in the evaluation set to be answered from a 10K document corpus. To produce a preliminary answer key, annotators were given the queries and corpus indexed by Google Desktop. Annotators were given 1 hour to find potential answers to each query. If no answers were found after 1 hour, the annotators were given a second hour to look for answers. For two queries, both of the form *Find treatments with an expected date to market of MM-YYYY*, even after two hours of searching the annotators were unable to find any answers.[6]

Annotator answers served as the initial gold-standard. Given this initial answer key, annotators reviewed system answers and aligned them with gold-standard answers. System output not aligned with the initial gold standard was assessed as correct or incorrect. We assume that the final gold-standard constitutes a complete answer key, and

[6] Evaluators wanted some queries with no answers.

are thus able to calculate recall for our system and for humans[7]. Because we had only one annotator for each query and because we assumed that any answer found by an annotator was correct, we could not estimate human precision on this task.

Answers can be specific named concepts (e.g. *Penicillin*) or generic descriptions (e.g. *drug, illness*). Given the sentence, *ACME produces a wide range of drugs including treatments for malaria and athletes foot*,' our reading system would extract the relations *responsibleForTreatment(drugs, ACME), possibleTreatment(drugs, malaria), possibleTreatment(drugs, athletes foot).* When a name was available in the document, annotators marked the answer as correct, but underspecified. We calculated precision and recall treating underspecified answers as incorrect and separately calculated precision and recall counting underspecified answers as correct. When treated as correct, there was less than a 0.05 absolute increase in both precision and recall. Unless otherwise specified, all scores reported here use the stricter condition which treats underspecified answers as incorrect.

We also evaluated extracting all information in a small document collection (here human search of the 10k documents does not play a role in finding answers). Individuals were asked to annotate every instance of the 5 relations in a set of 102 documents. Recall, Precision, and F were calculated by aligning system responses to the answer key. System answers that aligned are correct; those that did not are incorrect; and answers in the key that were not found by the system are misses. Unlike the question answering evaluation, this evaluation measures the ability to find every instance of a fact. If the gold standard includes 100 instances of *isPossibleTreatment(Penicillin, Strep Throat)*, recall will decrease for each instance missed. The "extraction" evaluation does not penalize systems for missing coreference.

6 Results

6.1 Class Detection

The recall, precision, and F1 for class detection using 10-fold cross validation of the ~1K annotated sentences appear in the 3-5th columns of Table 4. Given the amount of training, our results are lower than in Miller et al (2004) (an F1 of 90 with less than 25K words of training). Several factors could explain this: Finding boundaries and types for descriptions is more complex than for names in English.[8] Our classes, pharmaceutical substances and physiological conditions, may have been more difficult to learn. Our classes are less common in news reporting; as such, both word-class clusters and active learning may have been less effective. Finally, our evaluation was done on a 10-fold split of the active-learning selected data; bias in selecting the data could explain at least a part of our lower performance.

Type	**# in GS**	**Without Lists**			**With Lists**		
		R	**P**	**F**	**R**	**P**	**F**
Subst-D	789	77	85	80.8	78	85	81.3
Subst-N	410	70	82	75.5	77	81	78.9
Cond-D	427	72	78	74.9	72	77	74.4
Cond-N	963	80	87	83.4	84	83	83.5

Table 4: Cross Validation: Condition & Substance

We noticed that the system frequently reported country names to be substance-names. Surprisingly, we found that our well-trained name finder made the opposite mistake, occasionally reporting drugs as geo-political entities.

We incorporated lists of known substances and conditions to improve recall. Performance on the same cross-validation split is shown in the final three columns of Table 4. Incorporating the lists led to recall gains for both substance-name and condition-name. Because a false-alarm in class recognition only leads to an incorrect relation extraction if it appears in a context indicating a domain relation, false alarms of classes may be less important in the question answering and extraction evaluations.

6.2 Question Answering and Extraction

Figure 2 and Table 6 show system performance using only handwritten rules (HW), only learned patterns (L), and combining both (C). Figure 2 includes scores calculated with all of the systems' answers (in the dotted boxes), and with just those answers that were deemed useful (discussed be-

[7] The answer key may contain some answers that were found neither by the annotator nor by the systems described here, since the answer key includes answers pooled from other systems not reported in this paper. The system reported here was the highest performing of all those participating in the experiment. Furthermore, if a system answer is marked as correct, but underspecified, the specific answer is put in the key.

[8] English names are capitalized; person names have a typical form and are frequently signaled by titles; organization names frequently have clear signal words, such as Corp.

low). We include annotator recall. Handwritten patterns outperform learned patterns consistently with much higher recall. Encouragingly, however,

1. The combined system's recall and F-Score are noticeably higher for 3 of the relations.
2. The learned patterns generate answers not found by handwritten patterns.
3. The learned patterns have high precision.[9]

There is variation across the different relations. The two best performing relations *possibleTreatment*() and *studiesDisease*() have F1 more than twice as high as the two worst performing relations, *expectedDateToMarket()* and *hasSideEffect()*. This is primarily due to differences in recall.

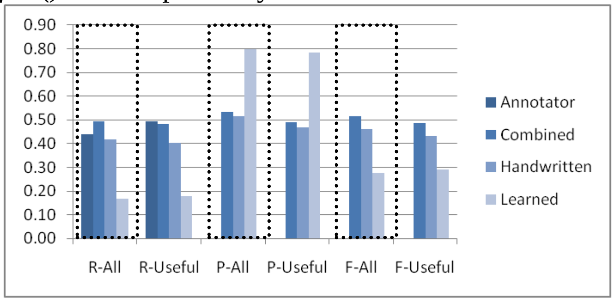

Figure 2: Overall Q/A Performance: All answers in dotted boxes; 'Useful Answers' unboxed

The combined system's recall (0.49), while low, is higher than that of the annotators (0.44). While hardly surprising that a machine can process information much more quickly than a person, it is encouraging that higher recall is achieved even with only one week's effort. In the context of our pooled answer-key, the relatively low recall of both the system and the annotator suggests that there was little overlap between the answers found by the annotator and those found by the system.

As already described, the system answers can include both specific references (e.g. *Prozac*) and more generic references (*the drug*). When a more specific answer is present in the document, generic references have been treated as incorrect. However, sometimes there is not a more specific reference; for example an article written before a drug has been released may never name the drug. Scores reported thus far treat such answers as correct. These answers would be useful when answering more complex queries. For example, given the sentence '*ACME spent 5 years developing a pill to treat the flu which it will release next week,'* extracting relations involving '*the pill'* would allow a system to answer questions that use multiple relations in the ontology to for example ask about *organizations developing treatments for the flu*, or *the expected date of release for ACME's drugs*. However, in our simple question answering framework such generic answers never convey novel information and thus were probably ignored by human annotators.

To measure the impact of treating these generic references as correct,[10] we did additional annotation on the correct answers, marking answers as 'useful' (specific) and 'not-useful' (generic). The unboxed bars in Figure 2 show performance when 'not-useful' answers are removed from the answer-key and the responses. For the four relations where there was a change Table 5 provides the relative change performance when only 'useful' answers are considered. The annotator's recall increases noticeably while the combined system's drops. This results in the overall recall of annotators surpassing that of the combined system.

Relation	**Recall**				**Precision**		
	A	**C**	**H**	**L**	**C**	**H**	**L**
possTreat	12	10	10	14	-10	-11	-3
respTreat	9	0	-5	8	-4	-4	-1
studyDis	12	-6	-9	13	-11	-13	0
hasSidEff	3	4	4	4	0	0	0
Total	11	-2	-4	6	-9	-10	-2

Table 5: Relative Change in Recall and Precision When Non-Useful Answers are Removed

Table 7 shows the total number of answers produced by annotators and by each system, as well as the percentage of queries with at least one correct answer for each system. For one relation *expectedDateOnMarket(),* the learned system did not find any answers. This relation had far fewer answers found by annotators and occurred far more rarely in the fully annotated extraction set (see Table 8). Anecdotally, extracting this relation frequently required co-referencing 'it' (e.g. "*It will be released in March 2011"*). Our heuristics for coreference of the new classes did not account for pronouns. Learning from such examples would require coreference during bootstrapping. Most likely, the learned system was unable to generate enough novel instances to continue bootstrapping

[9] The learned patterns' high precision is to be expected for two reasons. First, a few bad patterns were manually removed for each relation. More importantly, the learning algorithm strongly favors high precision patterns because it needs to maintain a seed set with low noise in order to learn effectively.

[10] Generic answers were treated as correct only if a more specific reference was not available in the document.

and was thus unable to learn the relation.

Relation Type (# Queries; # Correct Ans.)	Recall				Precision			F		
	A	C	HW	L	C	HW	L	C	HW	L
possTreatment (10;247)	0.27	0.63	0.50	0.34	0.51	0.47	0.83	0.56	0.48	0.48
respForTreat (15;134)	0.73	0.33	0.24	0.22	0.66	0.78	0.73	0.44	0.37	0.33
expectDateMarkt (11;60)	0.90	0.17	0.17	0.00	0.77	0.83	0.00	0.27	0.28	0
studiesDisease (13;292)	0.23	0.67	0.59	0.09	0.51	0.50	0.79	0.58	0.54	0.16
hasSideEffect (11;104)	0.80	0.10	0.13	0.02	0.83	0.70	1.00	0.17	0.23	0.04
Total (60;837)	0.44	0.49	0.42	0.17	0.53	0.52	0.80	0.51	0.46	0.28

Table 6: Question Answering Results by Relation Type

Relation Type	Total Number of Answers				% Queries with At Least 1 Corr. Ans			
	A	C	HW	L	A	C	HW	L
possTreatment	66	303	261	100	100.0%	90.0%	90.0%	90.0%
respForTreat	98	67	41	40	100.0%	66.7%	60.0%	60.0%
expectDateMarkt	54	13	12	0	72.7%	45.5%	45.5%	0.0%
studiesDisease	68	379	347	33	100.0%	61.5%	46.2%	46.2%
hasSideEffect	83	12	20	2	72.7%	36.4%	45.5%	18.2%
Total	369	774	681	175	90.0%	60.0%	56.7%	43.3%

Table 7: Number of Answers and Number of Queries Answered

Overall, the system did better on relations having more correct answers. Bootstrap learning has an easier time discovering new instances and new patterns when there are more examples to work with. Even a human pattern writer will have more examples to generalize from for common relations.

While *possibleTreatment()* and *hasSideEffect()* have similar F-scores, their performance is very different at the query level. The system was able to find at least one correct answer to every *possible-Treatment*() query; however only 72.7% of the *studiesDisease*() queries were answered.

Table 8 presents results from the extraction evaluation where a set of ~100 documents were annotated for all mentions of the 5 relations. Because every mention in the document set must be found, the system cannot rely on finding the easiest answers for common relations. The results in Table 8 are significantly lower than for the question answering tasks; yet some of the same trends are present. Handwritten rules outperform learned patterns. For at least some relations, the combination of the two improves performance. The three relations for which the learned system has the lowest performance on the question-answering task have the fewest instances annotated in the document set. Fewer instance in the large corpus make bootstrapping more difficult—the learner is less able to generate novel instances to expand its pattern set.

7 Discussion

7.1 Sources of Error

The most common source of error is pattern coverage. In the following figure, the system identified responsibleForTreatment(Janssen Pharmaceutical, Sporanox), but missed the corresponding relation between Novartis and Lamisil.

Sporanox is made by Janssen Pharmaceutica Inc., of Titusville, N.J. Lamisil is a product of Novartis Pharmaceuticals of East Hanover, N.J.

Relation Type	# Relations Found				Recall			Precision			F		
	GS	C	HW	L	C	HW	L	C	HW	L	C	HW	L
possibleTreatment	518	225	187	68	**0.15**	0.10	0.09	**0.34**	0.28	0.66	**0.21**	0.15	0.15
respForTreatment	387	101	77	36	**0.10**	0.08	0.05	**0.41**	0.40	0.50	**0.17**	0.13	0.08
expDateOnMarket	66	13	13	0	***0.06***	***0.06***	0.00	***0.31***	**0.31**	0.00	***0.10***	***0.10***	0.00
studiesDisease	136	95	91	4	0.08	***0.09***	0.00	0.12	**0.13**	0.00	0.10	**0.11**	0.00
hasSideEffect	256	26	25	2	***0.04***	***0.04***	0.00	0.39	0.40	0.50	***0.07***	***0.07***	0.01

Table 8: Extraction Results on the 102 Document Test Set Annotated for All Instances of the Relations

Missed class instances contribute to errors, sometimes originating in errors in tokenization (e.g. not removing the '_' in each drug name in a bulleted list of the form "*_Trovan, an antibiotic...; etc.)* However, many drug-names are simply missed:

> *Pfizer also hopes to introduce Pregabalin next year for treatment of neuropathic pain, epilepsy and anxiety…Other deals include co-promoting Rebif for multiple sclerosis with its discoverer, Serono, and marketing Aricept for Alzheimer's disease with its developer, Eisai Co.*

The system correctly identifies *Rebif* and *Aricept* as drugs, but misses *Pregabalin* and *Serono*. In both misses, the immediately preceding and following words provide little evidence that the word refers to a drug rather than some other product. Substance detection might be better served with a web-scale, list-learning approach like the doubly anchored patterns described in (Kozareva et al., 2008). Alternatively, our approach may need to be extended to include a larger context window.

7.2 Learned Patterns

One of the ways in which learned patterns supplement handwritten ones is learning highly specific surface-string patterns that are insensitive to errors in parsing. Figure 3 illustrates two examples of what appear to be easy cases of *possibleTreatment*(). Because the handwritten patterns are not exhaustive and make extensive use of syntactic structure, parse errors prevented the system based on handwritten rules from firing. Learned surface-string patterns were able to find these relations.

Even when the syntactic structure is correct, learned patterns capture expressions not common enough to have been noticed by the rule writer. For example, while the handwritten patterns included '*withdrew*' as a predicate indicating a company was responsible for a drug, they did not include '*pulled*.' By including '*pulled*', learned patterns extracted *responsibleForTreatment()* from '*American Home Products pulled Duract, a painkiller*.' Similarly, the learned patterns include an explicit pattern '*CONDITION drug called SUBSTANCE*', and thus extracted a *possibleTreatment()* relation from '*newly approved narcolepsy drug called modafinil*' without relying on the coreference component to link *drug* to *modafinil*.

Handwritten Patterns

Despite the examples above of successfully learned patterns, handwritten patterns perform significantly better. In the active-learning context used for these experiments, the handwritten rules also required *less* manual effort. This comparison is not entirely fair-- while learned patterns required more hours, supervising the bootstrapping algorithm requires no training. The handwritten patterns, in contrast, require a trained expert.

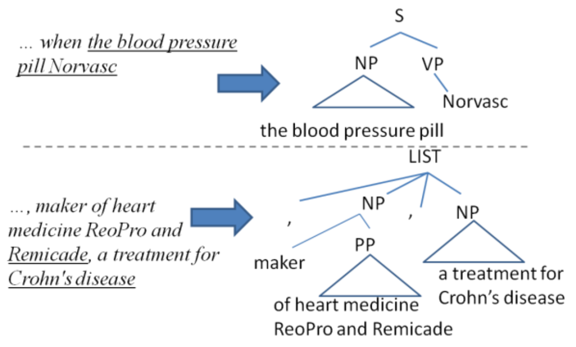
… when the blood pressure pill Norvasc
S
NP
VP
Norvasc
the blood pressure pill
LIST
…, maker of heart medicine ReoPro and Remicade, a treatment for Crohn's disease
NP
,
,
NP
maker
PP
of heart medicine ReoPro and Remicade
a treatment for Crohn's disease

Figure 3: Extractions Missed by Handwritten Rules & the Erroneous Parses that Hid them

While handwritten rules and learned patterns use the same language, they make use of it differently. The handwritten patterns group similar concepts together. A human pattern writer adds relevant synonyms, as well as words that are not synonymous but in the pattern context can be used interchangeably. In Figure 4, the handwritten patterns include three word-sets: (*patient**, *people*, *participant**); (*given*, *taken*, *took*, *using*); and (*report**, *experience**, *develop**, *suffer**). The '*' serves as a wild-card to further generalize a pattern. The word-sets in Figure 4 illustrate challenges for a learned system: the words are not synonyms, but rather are words that can be used to imply the relation.

A human pattern writer frequently generates new classes not in the domain ontology. In Figure 4, the circled patterns form a class of 'people taking a substance.' The handwritten patterns for *studiesDisease*() include classes targeting scientists and researchers. These classes are not necessarily triggered by nouns. Such classes allow the pattern writer to include complex patterns as in Figure 4 and to write relatively precise, but open-ended patterns such as: *if there is a single named-drug and a named, non-side-effect disease in the same sentence, the drug is a treatment for the disease.*

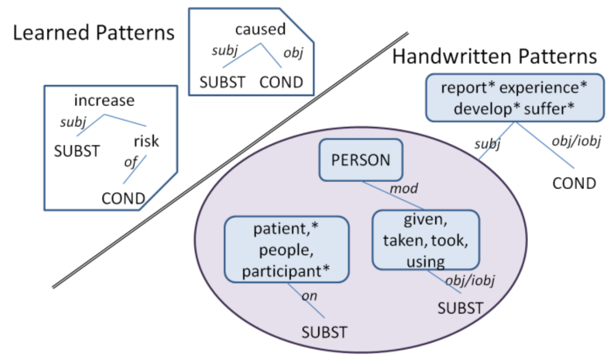

Figure 4: Learned and Handwritten Patterns for *hasSideEffect*()

A final difference between handwritten and learned patterns is the level of predicate-argument complexity used. In general, handwritten patterns account for larger spans of predicate argument structure while learned patterns tend to limit themselves to the connections between the arguments of the relation with minor extensions.

8 Conclusions and Lessons Learned

First, it is encouraging that the synthesis of learning algorithms and handwritten algorithms can achieve an F1 of 0.51 in a new domain in a week (<50 hours of effort). *Second*, it is exciting that so little training data is required: ~20 relation pairs out of context (~2.5 hours of effort) and ~6 hours of active learning for the new classes.

Third, the effectiveness of learning algorithms is still not competitive with handwritten patterns based on predicate-argument structure (~5 hours of effort on top of active learning for entities). Though the learned patterns have high precision (0.80 on average), recall is low (0.17) and varied greatly across the relations. Though the dominant factor in missing relations is pattern coverage, missing instances of classes contributed to low recall. Comparing learned patterns to manually written patterns, (1) synonyms or other lexical alternatives that a human pattern writer would include, (2) the creation of subclasses for argument types, and (3) the scope of patterns[11] are each major sources of the disparity in coverage. Research on learning approaches to raise recall without significant sacrifice in precision seems essential.

Fourth, despite the disparity in performance of learned versus manual patterns, and despite the low recall of learned patterns, the combined system's recall and F-Score are higher for three of the relations because the learned patterns generated answers not found by handwritten patterns. We found examples where highly specific, learned, surface-level patterns (lexical patterns) occasionally found information missed by handwritten patterns due to parsing errors or general low coverage.

Fifth, the effort for coreference was the most time-consuming, given that every new relation contained at least one of the new argument types. While we included this in our estimate of domain adaptation, the infrastructure we built is domain generic. Improving generic coreference will reduce domain specific effort in future.

Perhaps *most significant* of all, running a complete experiment from definition of the domain through creation of training data and measurement of end-to-end performance of the system can be completed in a month. The ability to rapidly, cheaply, and empirically measure the impact of extraction research could prove a significant spur to research across the board.

These experiments suggest three possible directions for improving the ability to quickly develop information extraction technology for a new set of relations: (1) reducing the amount of supervision provided to the bootstrap-learner; (2) improving the bootstrapping approach to reach the level of recall achieved by the human pattern writer eliminating the need for a trained expert during domain adaptation; and (3) focusing improvements to the bootstrapping approach on techniques that allow it to find more of the instances missed by the pattern writer, thus improving the accuracy of the hybrid system.

Acknowledgments

This work was supported, in part, by DARPA under AFRL Contract FA8750-09-C-179. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited) Thank you to the reviewers for your insightful comments and to Michelle Franchini for coordinating the assessment effort.

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.

[11] Learned patterns tend to focus on the structure that appears between the two arguments, rather than structure surrounding the left and right arguments.

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.

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

M. Freedman, E. Loper, E. Boschee, and R. Weischedel. Empirical Studies in Learning to Read. Proceedings of NAACL 2010 Workshop on Formalisms and Methodology for Learning by Reading, pp. 61-69, June 2010.

W. Li and A. McCallum. Rapid development of Hindi named entity recognition using conditional random fields and feature induction. Transactions on Asian Language Information Processing (TALIP), Volume 2 Issue 3 September, 2003.

R Grishman and B. Sundheim. Message Understanding Conference-6 : A Brief History", in COLING-96, *Proc . of the Int'l Conj. on Computational Linguistics*, 1996.

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

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056.

J. May, A. Brunstein, P. Natarajan, and R. Weischedel. Surprise! What's in a Cebuano or Hindi Name? Transactions on Asian Language Information Processing (TALIP), Volume 2 Issue 3 September, 2003.

D. Maynard, V. Tablan, K. Bontcheva, and H. Cunningham. Rapid customization of an information extraction system for a surprise language. Transactions on Asian Language Information Processing (TALIP), Volume 2 Issue 3 September, 2003.

S. Miller, J. Guinness, and A. Zamanian, "Name Tagging with Word Cluster and Discriminative Training", *Proceedings of HLT/NAACL* 2004, pp. 337-342, 2004

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).*

NIST, ACE 2007: http://www.itl.nist.gov/iad/mig/tests/ace/2007/software.html

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.

S. Sekine and R. Grishman. Hindi-English cross-lingual question-answering system. Transactions on Asian Language Information Processing (TALIP), Volume 2 Issue 3 September, 2003.

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