Learning Domain-Specific Information Extraction Patterns from the Web

Siddharth Patwardhan and Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
{sidd,riloff}@cs.utah.edu

Abstract

Many information extraction (IE) systems rely on manually annotated training data to learn patterns or rules for extracting information about events. Manually annotating data is expensive, however, and a new data set must be annotated for each domain. So most IE training sets are relatively small. Consequently, IE patterns learned from annotated training sets often have limited coverage. In this paper, we explore the idea of using the Web to automatically identify domain-specific IE patterns that were not seen in the training data. We use IE patterns learned from the MUC-4 training set as anchors to identify domain-specific web pages and then learn new IE patterns from them. We compute the *semantic affinity* of each new pattern to automatically infer the type of information that it will extract. Experiments on the MUC-4 test set show that these new IE patterns improved recall with only a small precision loss.

1 Introduction

Information Extraction (IE) is the task of identifying event descriptions in natural language text and extracting information related to those events. Many IE systems use extraction patterns or rules to identify the relevant information (Soderland et al., 1995; Riloff, 1996; Califf and Mooney, 1999; Soderland, 1999; Yangarber et al., 2000). Most of these systems use annotated training data to learn pattern matching rules based on lexical, syntactic, and/or semantic information. The learned patterns are then used to locate relevant information in new texts.

IE systems typically focus on information about events that are relevant to a specific domain, such as terrorism (Sundheim, 1992; Soderland et al., 1995; Riloff, 1996; Chieu et al., 2003), management succession (Sundheim, 1995; Yangarber et al., 2000), or job announcements (Califf and Mooney, 1999; Freitag and McCallum, 2000). Supervised learning systems for IE depend on domain-specific training data, which consists of texts associated with the domain that have been manually annotated with event information.

The need for domain-specific training data has several disadvantages. Because of the manual labor involved in annotating a corpus, and because a new corpus must be annotated for each domain, most annotated IE corpora are relatively small. Language is so expressive that it is practically impossible for the patterns learned from a relatively small training set to cover all the different ways of describing events. Consequently, the IE patterns learned from manually annotated training sets typically represent only a subset of the IE patterns that could be useful for the task. Many recent approaches in natural language processing (Yarowsky, 1995; Collins and Singer, 1999; Riloff and Jones, 1999; Nigam et al., 2000; Wiebe and Riloff, 2005) have recognized the need to use unannotated data to improve performance.

While the Web provides a vast repository of unannotated texts, it is non-trivial to identify texts that belong to a particular domain. The difficulty is that web pages are not specifically annotated with tags categorizing their content. Nevertheless, in this paper we look to the Web as a vast dynamic resource for domain-specific IE learning. Our approach exploits an existing set of IE patterns that were learned from annotated training data to automatically identify new, domain-specific texts from

Proceedings of the Workshop on Information Extraction Beyond The Document, pages 66–73,
Sydney, July 2006. ©2006 Association for Computational Linguistics

the Web. These web pages are then used for additional IE training, yielding a new set of domain-specific IE patterns. Experiments on the MUC-4 test set show that the new IE patterns improve coverage for the domain.

This paper is organized as follows. Section 2 presents the MUC-4 IE task and data that we use in our experiments. Section 3 describes how we create a baseline IE system from the MUC-4 training data. Section 4 describes the collection and preprocessing of potentially relevant web pages. Section 5 then explains how we use the IE patterns learned from the MUC-4 training set as anchors to learn new IE patterns from the web pages. We also compute the *semantic affinity* of each new pattern to automatically infer the type of information that it will extract. Section 6 shows experimental results for two types of extractions, victims and targets, on the MUC-4 test set. Finally, Section 7 compares our approach to related research, and Section 8 concludes with ideas for future work.

2 The MUC-4 IE Task and Data

The focus of our research is on the MUC-4 information extraction task (Sundheim, 1992), which is to extract information about terrorist events. The MUC-4 corpus contains 1700 stories, mainly news articles related to Latin American terrorism, and associated *answer key templates* containing the information that should be extracted from each story.

We focused our efforts on two of the MUC-4 *string* slots, which require textual extractions: human targets (victims) and physical targets. The MUC-4 data has proven to be an especially difficult IE task for a variety of reasons, including the fact that the texts are entirely in upper case, roughly 50% of the texts are irrelevant (i.e., they do not describe a relevant terrorist event), and many of the stories that are relevant describe multiple terrorist events that need to be teased apart. The best results reported across all string slots in MUC-4 were in the 50%-70% range for recall and precision (Sundheim, 1992), with most of the MUC-4 systems relying on heavily hand-engineered components. Chieu et al. (2003) recently developed a fully automatic template generator for the MUC-4 IE task. Their best system produced recall scores of 41%-44% with precision scores of 49%-51% on the TST3 and TST4 test sets.

3 Learning IE Patterns from a Fixed Training Set

As our baseline system, we created an IE system for the MUC-4 terrorism domain using the AutoSlog-TS extraction pattern learning system (Riloff, 1996; Riloff and Phillips, 2004), which is freely available for research use. AutoSlog-TS is a weakly supervised learner that requires two sets of texts for training: texts that are relevant to the domain and texts that are irrelevant to the domain. The MUC-4 data includes relevance judgments (implicit in the answer keys), which we used to partition our training set into relevant and irrelevant subsets.

AutoSlog-TS' learning process has two phases. In the first phase, syntactic patterns are applied to the training corpus in an exhaustive fashion, so that extraction patterns are generated for (literally) every lexical instantiation of the patterns that appears in the corpus. For example, the syntactic pattern *"<subj> PassVP"* would generate extraction patterns for all verbs that appear in the corpus in a passive voice construction. The subject of the verb will be extracted. In the terrorism domain, some of these extraction patterns might be: *"<subj> PassVP(murdered)"* and *"<subj> PassVP(bombed)."* These would match sentences such as: "the mayor was murdered", and "the embassy and hotel were bombed". Figure 1 shows the 17 types of extraction patterns that AutoSlog-TS currently generates. PassVP refers to passive voice verb phrases (VPs), ActVP refers to active voice VPs, InfVP refers to infinitive VPs, and AuxVP refers to VPs where the main verb is a form of "to be" or "to have". Subjects (subj), direct objects (dobj), PP objects (np), and possessives can be extracted by the patterns.

In the second phase, AutoSlog-TS applies all of the generated extraction patterns to the training corpus and gathers statistics for how often each pattern occurs in relevant versus irrelevant texts. The extraction patterns are subsequently ranked based on their association with the domain, and then a person manually reviews the patterns, deciding which ones to keep[1] and assigning thematic roles to them. We manually defined selectional restrictions for each slot type (victim and target)

[1] Typically, many patterns are strongly associated with the domain but will not extract information that is relevant to the IE task. For example, in this work we only care about patterns that will extract victims and targets. Patterns that extract other types of information are not of interest.

Pattern Type	Example Pattern
<subj> PassVP	<victim> was murdered
<subj> ActVP	<perp> murdered
<subj> ActVP Dobj	<weapon> caused damage
<subj> ActInfVP	<perp> tried to kill
<subj> PassInfVP	<weapon> was intended to kill
<subj> AuxVP Dobj	<victim> was casualty
<subj> AuxVP Adj	<victim> is dead
ActVP <dobj>	bombed <target>
InfVP <dobj>	to kill <victim>
ActInfVP <dobj>	planned to bomb <target>
PassInfVP <dobj>	was planned to kill <victim>
Subj AuxVP <dobj>	fatality is <victim>
NP Prep <np>	attack against <target>
ActVP Prep <np>	killed with <weapon>
PassVP Prep <np>	was killed with <weapon>
InfVP Prep <np>	to destroy with <weapon>
<possessive> NP	<victim>'s murder

Figure 1: AutoSlog-TS' pattern types and sample IE patterns

and then automatically added these to each pattern when the role was assigned.

On our training set, AutoSlog-TS generated 40,553 distinct extraction patterns. A person manually reviewed all of the extraction patterns that had a score $\geq$ 0.951 and frequency $\geq$ 3. This score corresponds to AutoSlog-TS' RlogF metric, described in (Riloff, 1996). The lowest ranked patterns that passed our thresholds had at least 3 relevant extractions out of 5 total extractions. In all, 2,808 patterns passed the thresholds. The reviewer ultimately decided that 396 of the patterns were useful for the MUC-4 IE task, of which 291 were useful for extracting victims and targets.

4 Data Collection

In this research, our goal is to automatically learn IE patterns from a large, domain-independent text collection, such as the Web. The billions of freely available documents on the World Wide Web and its ever-growing size make the Web a potential source of data for many corpus-based natural language processing tasks. Indeed, many researchers have recently tapped the Web as a data-source for improving performance on NLP tasks (e.g., Resnik (1999), Ravichandran and Hovy (2002), Keller and Lapata (2003)). Despite these successes, numerous problems exist with collecting data from the Web, such as web pages containing information that is not free text, including advertisements, embedded scripts, tables, captions, etc. Also, the documents cover many genres, and it is not easy to identify documents of a particular genre or domain. Additionally, most of the documents are in HTML, and some amount of processing is required to extract the free text. In the following subsections we describe the process of collecting a corpus of terrorism-related CNN news articles from the Web.

4.1 Collecting Domain-Specific Texts

Our goal was to automatically identify and collect a set of documents that are similar in domain to the MUC-4 terrorism text collection. To create such a corpus, we used hand-crafted queries given to a search engine. The queries to the search engine were manually created to try to ensure that the majority of the documents returned by the search engine would be terrorism-related. Each query consisted of two parts: (1) the name of a terrorist organization, and (2) a word or phrase describing a terrorist action (such as *bombed*, *kidnapped*, etc.). The following lists of 5 terrorist organizations and 16 terrorist actions were used to create search engine queries:

> **Terrorist organizations:** *Al Qaeda, ELN, FARC, HAMAS, IRA*
>
> **Terrorist actions:** *assassinated, assassination, blew up, bombed, bombing, bombs, explosion, hijacked, hijacking, injured, kidnapped, kidnapping, killed, murder, suicide bomber, wounded.*

We created a total of 80 different queries representing each possible combination of a terrorist organization and a terrorist action.

We used the *Google*[2] search engine with the help of the freely available *Google API*[3] to locate the texts on the Web. To ensure that we retrieved only CNN news articles, we restricted the search to the domain "`cnn.com`" by adding the "`site:`" option to each of the queries. We also restricted the search to English language documents by initializing the API with the `lang_en` option. We deleted documents whose URLs contained the word "transcript" because most of these were transcriptions of CNN's TV shows and were stylistically very different from written text. We ran the 80 queries twice, once in December 2005 and once in April 2005, which produced 3,496 documents and 3,309 documents, respectively. After removing duplicate articles, we were left

[2] http://www.google.com

[3] http://www.google.com/apis

with a total of 6,182 potentially relevant terrorism articles.

4.2 Processing the Texts

The downloaded documents were all HTML documents containing HTML tags and JavaScript intermingled with the news text. The CNN webpages typically also contained advertisements, text for navigating the website, headlines and links to other stories. All of these things could be problematic for our information extraction system, which was designed to process narrative text using a shallow parser. Thus, simply deleting all HTML tags on the page would not have given us natural language sentences. Instead, we took advantage of the uniformity of the CNN web pages to "clean" them and extract just the sentences corresponding to the news story.

We used a tool called *HTMLParser*[4] to parse the HTML code, and then deleted all nodes in the HTML parse trees corresponding to tables, comments, and embedded scripts (such as JavaScript or VBScript). The system automatically extracted news text starting from the headline (embedded in an *H1* HTML element) and inferred the end of the article text using a set of textual clues such as *"Feedback:"*, *"Copyright 2005"*, *"contributed to this report"*, etc. In case of any ambiguity, all of the text on the web page was extracted.

The size of the text documents ranged from 0 bytes to 255 kilobytes. The empty documents were due to dead links that the search engine had indexed at an earlier time, but which no longer existed. Some extremely small documents also resulted from web pages that had virtually no free text on them, so only a few words remained after the HTML had been stripped. Consequently, we removed all documents less than 10 bytes in size. Upon inspection, we found that many of the largest documents were political articles, such as political party platforms and transcriptions of political speeches, which contained only brief references to terrorist events. To prevent the large documents from skewing the corpus, we also deleted all documents over 10 kilobytes in size. At the end of this process we were left with a CNN terrorism news corpus of 5,618 documents, each with an average size of about 648 words. In the rest of the paper we will refer to these texts as "the CNN terrorism web pages".

5 Learning Domain-Specific IE Patterns from Web Pages

Having created a large domain-specific corpus from the Web, we are faced with the problem of identifying the useful extraction patterns from these new texts. Our basic approach is to use the patterns learned from the fixed training set as *seed patterns* to identify sentences in the CNN terrorism web pages that describe a terrorist event. We hypothesized that extraction patterns occurring in the same sentence as a seed pattern are likely to be associated with terrorism.

Our process for learning new domain-specific IE patterns has two phases, which are described in the following sections. Section 5.1 describes how we produce a ranked list of candidate extraction patterns from the CNN terrorism web pages. Section 5.2 explains how we filter these patterns based on the *semantic affinity* of their extractions, which is a measure of the tendency of the pattern to extract entities of a desired semantic category.

5.1 Identifying Candidate Patterns

The first goal was to identify extraction patterns that were relevant to our domain: terrorist events. We began by exhaustively generating every possible extraction pattern that occurred in our CNN terrorism web pages. We applied the AutoSlog-TS system (Riloff, 1996) to the web pages to automatically generate all lexical instantiations of patterns in the corpus. Collectively, the resulting patterns were capable of extracting every noun phrase in the CNN collection. In all, 147,712 unique extraction patterns were created as a result of this process.

Next, we computed the statistical correlation of each extraction pattern with the seed patterns based on the frequency of their occurrence in the same sentence. IE patterns that never occurred in the same sentence as a seed pattern were discarded. We used Pointwise Mutual Information (PMI) (Manning and Schütze, 1999; Banerjee and Pedersen, 2003) as the measure of statistical correlation. Intuitively, an extraction pattern that occurs more often than chance in the same sentence as a seed pattern will have a high PMI score.

The 147,712 extraction patterns acquired from the CNN terrorism web pages were then ranked by their PMI correlation to the seed patterns. Table 1 lists the most highly ranked patterns. Many of these patterns do seem to be related to terrorism,

[4] http://htmlparser.sourceforge.net

<subj> killed sgt	<subj> destroyed factories
<subj> burned flag	explode after <np>
sympathizers of <np>	<subj> killed heir
<subj> kills bystanders	<subj> shattered roof
rescued within <np>	fled behind <np>

Table 1: Examples of candidate patterns that are highly correlated with the terrorism seed patterns

but many of them are not useful to our IE task (for this paper, identifying the victims and physical targets of a terrorist attack). For example, the pattern *"explode after <np>"* will not extract victims or physical targets, while the pattern *"sympathizers of <np>"* may extract people but they would not be the *victims* of an attack. In the next section, we explain how we filter and re-rank these candidate patterns to identify the ones that are directly useful to our IE task.

5.2 Filtering Patterns based upon their Semantic Affinity

Our next goal is to filter out the patterns that are not useful for our IE task, and to automatically assign the correct slot type (victim or target) to the ones that are relevant. To automatically determine the mapping between extractions and slots, we define a measure called *semantic affinity*. The semantic affinity of an extraction pattern to a semantic category is a measure of its tendency to extract NPs belonging to that semantic category. This measure serves two purposes:

(a) It allows us to filter out candidate patterns that do not have a strong semantic affinity to our categories of interest.

(b) It allows us to define a mapping between the extractions of the candidate patterns and the desired slot types.

We computed the semantic affinity of each candidate extraction pattern with respect to six semantic categories: *target*, *victim*, *perpetrator*, *organization*, *weapon* and *other*. Targets and victims are our categories of interest. Perpetrators, organizations, and weapons are common semantic classes in this domain which could be "distractors". The other category is a catch-all to represent all other semantic classes. To identify the semantic class of each noun phrase, we used the Sundance package (Riloff and Phillips, 2004), which is a freely available shallow parser that uses dictionaries to assign semantic classes to words and phrases.

We counted the frequencies of the semantic categories extracted by each candidate pattern and applied the RLogF measure used by AutoSlog-TS (Riloff, 1996) to rank the patterns based on their affinity for the target and victim semantic classes. For example, the semantic affinity of an extraction pattern for the target semantic class would be calculated as:

$$\text{affinity}_{pattern} = \frac{f_{target}}{f_{all}} \cdot \log_2 f_{target} \qquad (1)$$

where f_{target} is the number of target semantic class extractions and $f_{all} = f_{target} + f_{victim} + f_{perp} + f_{org} + f_{weapon} + f_{other}$. This is essentially a probability $P(target)$ weighted by the log of the frequency.

We then used two criteria to remove patterns that are not strongly associated with a desired semantic category. If the semantic affinity of a pattern for category C was (1) greater than a threshold, and (2) greater than its affinity for the *other* category, then the pattern was deemed to have a semantic affinity for category C. Note that we intentionally allow for a pattern to have an affinity for more than one semantic category (except for the catch-all *other* class) because this is fairly common in practice. For example, the pattern *"attack on <np>"* frequently extracts both targets (e.g., *"an attack on the U.S. embassy"*) and victims (e.g., *"an attack on the mayor of Bogota"*). Our hope is that such a pattern would receive a high semantic affinity ranking for both categories.

Table 2 shows the top 10 high frequency ($freq \geq 50$) patterns that were judged to have a strong semantic affinity for the target and victim categories. There are clearly some incorrect entries (e.g., *"<subj> fired missiles"* is more likely to identify perpetrators than targets), but most of the patterns are indeed good extractors for the desired categories. For example, *"fired into <np>"*, *"went off in <np>"*, and *"car bomb near <np>"* are all good patterns for identifying targets of a terrorist attack. In general, the semantic affinity measure seemed to do a reasonably good job of filtering patterns that are not relevant to our task, and identifying patterns that are useful for extracting victims and targets.

6 Experiments and Results

Our goal has been to use IE patterns learned from a fixed, domain-specific training set to automatically learn additional IE patterns from a large,

Target Patterns	Victim Patterns
<subj> fired missiles	wounded in <np>
missiles at <np>	<subj> was identified
bomb near <np>	wounding <dobj>
fired into <np>	<subj> wounding
died on <np>	identified <dobj>
went off in <np>	<subj> identified
car bomb near <np>	including <dobj>
exploded outside <np>	<subj> ahmed
gunmen on <np>	<subj> lying
killed near <np>	<subj> including

Table 2: Top 10 high-frequency target and victim patterns learned from the Web

domain-independent text collection, such as the Web. Although many of the patterns learned from the CNN terrorism web pages look like good extractors, an open question was whether they would actually be useful for the original IE task. For example, some of the patterns learned from the CNN web pages have to do with beheadings (e.g., "*beheading of <np>*" and "*beheaded <np>*"), which are undeniably good victim extractors. But the MUC-4 corpus primarily concerns Latin American terrorism that does not involve beheading incidents. In general, the question is whether IE patterns learned from a large, diverse text collection can be valuable for a specific IE task above and beyond the patterns that were learned from the domain-specific training set, or whether the newly learned patterns will simply not be applicable. To answer this question, we evaluated the newly learned IE patterns on the MUC-4 test set.

The MUC-4 data set is divided into 1300 development (DEV) texts, and four test sets of 100 texts each (TST1, TST2, TST3, and TST4).[5] All of these texts have associated answer key templates. We used 1500 texts (DEV+TST1+TST2) as our training set, and 200 texts (TST3+TST4) as our test set.

The IE process typically involves extracting information from individual sentences and then mapping that information into answer key templates, one template for each terrorist event described in the story. The process of template generation requires discourse processing to determine how many events took place and which facts correspond to which event. Discourse processing and template generation are not the focus of this paper. Our research aims to produce a larger set of extraction patterns so that more information will be extracted from the sentences, before discourse analysis would begin. Consequently, we evaluate the performance of our IE system at that stage: after extracting information from sentences, but before template generation takes place. This approach directly measures how well we are able to improve the coverage of our extraction patterns for the domain.

6.1 Baseline Results on the MUC-4 IE Task

The AutoSlog-TS system described in Section 3 used the MUC-4 training set to learn 291 target and victim IE patterns. These patterns produced 64% recall with 43% precision on the targets, and 50% recall with 52% precision on the victims.[6]

These numbers are not directly comparable to the official MUC-4 scores, which evaluate template generation, but our recall is in the same ballpark. Our precision is lower, but this is to be expected because we do not perform discourse analysis.[7] These 291 IE patterns represent our *baseline* IE system that was created from the MUC-4 training data.

6.2 Evaluating the Newly Learned Patterns

We used all 396 terrorism extraction patterns learned from the MUC-4 training set[8] as seeds to identify relevant text regions in the CNN terrorism web pages. We then produced a ranked list of new terrorism IE patterns using a semantic affinity cutoff of 3.0. We selected the top N patterns from the ranked list, with N ranging from 50 to 300, and added these N patterns to the baseline system.

Table 3 lists the recall, precision and F-measure for the increasingly larger pattern sets. For the tar-

[5] The DEV texts were used for development in MUC-3 and MUC-4. The TST1 and TST2 texts were used as test sets for MUC-3 and then as development texts for MUC-4. The TST3 and TST4 texts were used as the test sets for MUC-4.

[6] We used a *head noun* scoring scheme, where we scored an extraction as correct if its head noun matched the head noun in the answer key. This approach allows for different leading modifiers in an NP as long as the head noun is the same. For example, "armed men" will successfully match "5 armed men". We also discarded pronouns (they were not scored at all) because our system does not perform coreference resolution.

[7] Among other things, discourse processing merges seemingly disparate extractions based on coreference resolution (e.g., "the guerrillas" may refer to the same people as "the armed men") and applies task-specific constraints (e.g., the MUC-4 task definition has detailed rules about exactly what types of people are considered to be terrorists).

[8] This included not only the 291 target and victim patterns, but also 105 patterns associated with other types of terrorism information.

	Targets			Victims		
	Precision	Recall	F-measure	Precision	Recall	F-measure
baseline	0.425	0.642	0.511	0.498	0.517	0.507
50+baseline	0.420	0.642	0.508	0.498	0.517	0.507
100+baseline	0.419	0.650	0.510	0.496	0.521	0.508
150+baseline	0.415	0.650	0.507	0.480	0.521	0.500
200+baseline	0.412	0.667	0.509	0.478	0.521	0.499
250+baseline	0.401	0.691	0.507	0.478	0.521	0.499
300+baseline	0.394	0.691	0.502	0.471	0.542	0.504

Table 3: Performance of new IE patterns on MUC-4 test set

get slot, the recall increases from 64.2% to 69.1% with a small drop in precision. The F-measure drops by about 1% because recall and precision are less balanced. But we gain more in recall (+5%) than we lose in precision (-3%). For the victim patterns, the recall increases from 51.7% to 54.2% with a similar small drop in precision. The overall drop in the F-measure in this case is negligible. These results show that our approach for learning IE patterns from a large, diverse text collection (the Web) can indeed improve coverage on a domain-specific IE task, with a small decrease in precision.

7 Related Work

Unannotated texts have been used successfully for a variety of NLP tasks, including named entity recognition (Collins and Singer, 1999), subjectivity classification (Wiebe and Riloff, 2005), text classification (Nigam et al., 2000), and word sense disambiguation (Yarowsky, 1995). The Web has become a popular choice as a resource for large quantities of unannotated data. Many research ideas have exploited the Web in unsupervised or weakly supervised algorithms for natural language processing (e.g., Resnik (1999), Ravichandran and Hovy (2002), Keller and Lapata (2003)).

The use of unannotated data to improve information extraction is not new. Unannotated texts have been used for weakly supervised training of IE systems (Riloff, 1996) and in bootstrapping methods that begin with seed words or patterns (Riloff and Jones, 1999; Yangarber et al., 2000). However, those previous systems rely on pre-existing domain-specific corpora. For example, EXDISCO (Yangarber et al., 2000) used Wall Street Journal articles for training. AutoSlog-TS (Riloff, 1996) and Meta-bootstrapping (Riloff and Jones, 1999) used the MUC-4 training texts. Meta-bootstrapping was also trained on web pages, but the "domain" was corporate relationships so domain-specific web pages were easily identified simply by gathering corporate web pages.

The KNOWITALL system (Popescu et al., 2004) also uses unannotated web pages for information extraction. However, this work is quite different from ours because KNOWITALL focuses on extracting domain-independent relationships with the aim of extending an ontology. In contrast, our work focuses on using the Web to augment a domain-specific, event-oriented IE system with new, automatically generated domain-specific IE patterns acquired from the Web.

8 Conclusions and Future Work

We have shown that it is possible to learn new extraction patterns for a domain-specific IE task by automatically identifying domain-specific web pages using seed patterns. Our approach produced a 5% increase in recall for extracting targets and a 3% increase in recall for extracting victims of terrorist events. Both increases in recall were at the cost of a small loss in precision.

In future work, we plan to develop improved ranking methods and more sophisticated semantic affinity measures to further improve coverage and minimize precision loss. Another possible avenue for future work is to embed this approach in a bootstrapping mechanism so that the most reliable new IE patterns can be used to collect additional web pages, which can then be used to learn more IE patterns in an iterative fashion. Also, while most of this process is automated, some human intervention is required to create the search queries for the document collection process, and to generate the seed patterns. We plan to look into techniques to automate these manual tasks as well.

Acknowledgments

This research was supported by NSF Grant IIS-0208985 and the Institute for Scientific Computing Research and the Center for Applied Scientific Computing within Lawrence Livermore National Laboratory.

References

S. Banerjee and T. Pedersen. 2003. The Design, Implementation, and Use of the Ngram Statistics Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 370–381, Mexico City, Mexico, February.

M. Califf and R. Mooney. 1999. Relational Learning of Pattern-matching Rules for Information Extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, Orlando, FL, July.

H. Chieu, H. Ng, and Y. Lee. 2003. Closing the Gap: Learning-Based Information Extraction Rivaling Knowledge-Engineering Methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 216–223, Sapporo, Japan, July.

M. Collins and Y. Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, College Park, MD, June.

D. Freitag and A. McCallum. 2000. Information Extraction with HMM Structures Learned by Stochastic Optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 584–589, Austin, TX, August.

F. Keller and M. Lapata. 2003. Using the Web to Obtain Frequencies for Unseen Bigrams. *Computational Linguistics*, 29(3):459–484, September.

C. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2-3):103–134, May.

A. Popescu, A. Yates, and O. Etzioni. 2004. Class Extraction from the World Wide Web. In Ion Muslea, editor, *Adaptive Text Extraction and Mining: Papers from the 2004 AAAI Workshop*, pages 68–73, San Jose, CA, July.

D. Ravichandran and E. Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 41–47, Philadelphia, PA, July.

P. Resnik. 1999. Mining the Web for Bilingual Text. In *Proceedings of the 37th meeting of the Association for Computational Linguistics*, pages 527–534, College Park, MD, June.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 474–479, Orlando, FL, July.

E. Riloff and W. Phillips. 2004. An Introduction to the Sundance and AutoSlog Systems. Technical Report UUCS-04-015, School of Computing, University of Utah.

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Articial Intelligence*, pages 1044–1049, Portland, OR, August.

S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. 1995. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1319, Montreal, Canada, August.

S. Soderland. 1999. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272, February.

B. Sundheim. 1992. Overview of the Fourth Message Understanding Evaluation and Conference. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 3–21, McLean, VA, June.

B. Sundheim. 1995. Overview of the Results of the MUC-6 Evaluation. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 13–31, Columbia, MD, November.

J. Wiebe and E. Riloff. 2005. Creating Subjective and Objective Sentence Classifiers from Unannotated Texts. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 486–497, Mexico City, Mexico, February.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 940–946, Saarbrücken, Germany, August.

D. Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA, June.