Adapting Open Information Extraction to Domain-Specific Relations

Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni

Information extraction (IE) can identify a set of relations from free text to support question answering (QA). Until recently, IE systems were domain specific and needed a combination of manual engineering and supervised learning to adapt to each target domain. A new paradigm, Open IE, operates on large text corpora without any manual tagging of relations, and indeed without any prespecified relations. Due to its open-domain and open-relation nature, Open IE is purely textual and is unable to relate the surface forms to an ontology, if known in advance. We explore the steps needed to adapt Open IE to a domain-specific ontology and demonstrate our approach of mapping domain-independent tuples to an ontology using domains from the DARPA Machine Reading Project. Our system achieves precision over 0.90 from as few as eight training examples for an NFL-scoring domain.

Information extraction (IE) can identify a set of relations from free text to support question answering (QA). Until recently, IE systems were domain specific and needed a combination of manual engineering and supervised learning to adapt to each target domain. A new paradigm, Open IE, operates on large text corpora without any manual tagging of relations, and indeed without any prespecified relations. Due to its open-domain and open-relation nature, Open IE is purely textual and is unable to relate the surface forms to an ontology, if known in advance. We explore the steps needed to adapt Open IE to a domain-specific ontology and demonstrate our approach of mapping domain-independent tuples to an ontology using domains from the DARPA Machine Reading Project. Our system achieves precision over 0.90 from as few as eight training examples for an NFL-scoring domain.

Human reading relies on a mixture of domain-independent processes and domain-specific knowledge. Until recently, information extraction has leaned heavily on domain knowledge, which requires either manual engineering or manual tagging of examples (Miller et al. 1998; Soderland 1999; Culotta, McCallum, and Betz 2006). Semisupervised approaches (Riloff and Jones 1999, Agichtein and Gravano 2000, Rosenfeld and Feldman 2007) require only a small amount of hand-annotated training, but require this for every relation of interest. This still presents a knowledge engineering bottleneck, when one considers the unbounded number of relations in a diverse corpus such as the web. Shinyama and Sekine (2006) explored unsupervised relation discovery using a clustering algorithm with good precision, but limited scalability.

The KnowItAll research group is a pioneer of a new paradigm, Open IE (Banko et al. 2007, Banko and Etzioni 2008), that operates in a totally domain-independent manner and at web scale. An Open IE system makes a single pass over its corpus and extracts a diverse set of relational tuples without requiring any relation-specific human input. Open IE is ideally suited to corpora such as the web, where the target relations are not known in advance and their number is massive.
This is a challenging task, because an Open IE system has to locate both the set of entities believed to participate in a relation and the salient textual cues that indicate the relation among them.

TEXTRUNNER (Banko et al. 2007, Banko and Etzioni 2008) is an implemented Open IE system that operates at web scale, with lightweight processing that is linear in the number of documents and constant in the number of relations. TEXTRUNNER uses a part-of-speech tagger and noun phrase (NP) chunker and then identifies words that denote a relation between a pair of NPs in a sentence using a conditional random field (CRF) (Lafferty, McCallum, and Pereira 2001). Identifying the relation phrase is treated as a sequence-labeling problem — the CRFs are undirected graphical models trained to maximize the conditional probability that a sequence of words form a plausible relation.

TEXTRUNNER can produce a large corpus of generic relations in textual, surface forms. However, additional processing is required to match these textual relations with a domain-specific ontology. Matching to a known ontology is necessary to facilitate complex question answering, deep inference over the data, and combining knowledge bases generated by several systems in which the ontology may act as a common representation. In this article we explore what is involved in this further processing of Open IE relations. We ground this exploration in domains from the Defense Advanced Research Project Agency (DARPA) Machine Reading Project, which was launched in 2009.

Over the course of the 5-year Machine Reading Project, there will be two new domains per year. Of those, the first domain is NFL-scoring, in which systems must extract final scores of American football games, detect the winners and losers of the game, the score of each team, the game date, and similar relations. Figure 1 shows the desired output for a sample sentence from this domain.

The second domain is intelligence community (IC) scenarios, in which systems must extract information about agent, target, location and date of attacks, killing, injuring, and bombing, along with other information about persons and organizations. This domain is more challenging than the NFL-scoring domain, where there is a closed class of team names and regularity in the phrases that refer to a game or a score. In the IC domain, there is great variation in the phrases that can refer to agents or targets of attacks: any person, group of people, organization, or nation can be an agent; any person, group, organization, city, or physical structure can be a target.
The focus of this work is on creating a quickly reconfigurable system that takes Open IE extractions, a small amount of manual effort, and very limited training data — the output is instances grounded in the given ontology at high precision and recall. Our key ideas include: (1) domain-specific class recognizers built through minimal manual effort, (2) learning rules for relation extraction based on limited training data, and (3) active learning over learned rules to increase precision and recall. Figure 2 shows the system architecture.

We present a formal evaluation of our system on the NFL-scoring domain, where results indicate the value of these techniques, often yielding precision over 0.90 at reasonable coverage. We discuss preliminary work in adapting those techniques to the rich diversity in the IC domain.

Domain Adaptation

The goal of the DARPA Machine Reading Project is to design general-purpose readers that can be quickly adapted to particular domains. The formal evaluation of system performance will be question answering in terms of domain-specific relations. In the first year, research teams are given new domains in advance — by year three the goal is to handle new domains on the fly.

Figure 1 shows a sentence from an NFL football text, along with tuples extracted by the TEXTRUNNER Open IE system. Below that are the desired relations in the NFL-scoring ontology as defined by the DARPA Machine Reading Project organizers. All the required information is in the Open IE tuples, but substantial postprocessing is needed to derive the domain-specific relations from the tuples. Much of the information relevant to the domain is contained in subphrases of the NPs, while the relation phrases “cruised to” or “over” convey little information.

To meet the challenge of portability, an IE system must rely primarily on domain-independent processing. This allows it to keep the domain-specific knowledge to a minimum level that can be learned from a small amount of training, perhaps as few as 10 examples of each domain relation. Our combination of Open IE and lightly trained domain adaptation is a step in this direction. Figure 2 shows our system architecture.

The initial release of the NFL-scoring domain is
Implicitly defined by 27 sentences in which 13 relations were tagged exhaustively. Figure 3 lists the relations that had at least eight training examples. In this article we explore how well an Open IE system can adapt to a domain ontology with such a minimal tagging effort.

Figure 3. Relations for the NFL-Scoring Domain with the Number of Training Examples.

<table>
<thead>
<tr>
<th>NFL Relation</th>
<th>Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>GameWinner(NFLGame,NFLTeam)</td>
<td>15</td>
</tr>
<tr>
<td>GameLoser(NFLGame,NFLTeam)</td>
<td>14</td>
</tr>
<tr>
<td>TeamFinalScore(NFLGame,FinalScore)</td>
<td>29</td>
</tr>
<tr>
<td>TeamScoringAll(NNFLTeam,FinalScore)</td>
<td>46</td>
</tr>
<tr>
<td>GameDate(NFLGame,Date)</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 4. A High-Recall Version of TEXTRUNNER.

An earlier version of TEXTRUNNER extracted two high-confidence tuples from the sentence in Figure 1. A high-recall extension to TEXTRUNNER was necessary to find tuples that included the second team, “the 49ers.”

Sentence:
“Blake threw three touchdown passes as New Orleans opened up a 28–0 halftime lead and cruised to a 31–15 victory over the 49ers.”

High Confidence Tuples:
(New Orleans, cruised to, a 31–15 victory) 0.816
(Blake, threw, three touchdown passes) 0.833

Additional tuples:
(a 31–15 victory, over, the 49ers)
(a 28–0 halftime lead, cruised to, a 31–15 victory)
(New Orleans, opened, a 28–0 halftime lead)
(three touchdown passes, as, New Orleans)

Machine Reading Project QA task is in terms of formal relation types and argument types that do not resemble surface forms.

Extending TEXTRUNNER’s Recall

The version of TEXTRUNNER described by M. Banko and O. Etzioni (Banko et al. 2007, Banko and Etzioni 2008) considers relations between each pair of NPs in a sentence but extracts only those with highest confidence. We are currently working on a new version of TEXTRUNNER that substantially increases its recall, but the problem will still remain that portions of a sentence may not be covered by Open IE relations.

To compensate for this, we created a high-recall version of TEXTRUNNER by adding a tuple from each pair of adjacent NPs in the sentence. For many of these the predicate is simply a preposition between adjacent NPs, such as (a 31–15 victory over the 49ers) shown in figure 4. This is not sufficient evidence for a high-confidence tuple in Open IE, but these additional tuples can result in high-confidence domain relations when they contain terms relevant to the domain.

Mapping to Domain Relations

In contrast to Open IE in which the tuple arguments and predicates are taken directly from text, a domain ontology may have relations and argument types that do not resemble surface forms. Where Open IE treats entire noun phrases as argument values, arguments for domain relations may be filled by subphrases within an NP. Our goal is to learn a mapping of surface forms to domain relations such as TeamFinalScore(NFLGame,FinalScore) or Bombing(Agent,Target) from a small number of training examples. We have implemented this as a two-step process:

Step One. Learn a set of class recognizers and use them to label terms in the tuples. This can include domain-independent semantic tagging and named entity recognizers (NERs) to support argument type constraints.

Step Two. Learn a mapping to domain relations from tuples that have been processed by the class recognizers.

Figure 3 shows our architecture for a system that maps TEXTRUNNER’s domain-independent tuples to relations in a particular domain ontology. The Open IE extractor takes a set of documents as input and produces a set of relational tuples. These tuples are annotated by concept recognizers for a domain and then a set of relation-mapping rules operate on the tuples to create domain-specific relations.

Class Recognizers. The first step in domain adaptation is to identify terms or phrases in the tuples as instances of domain-specific classes. We create a class recognizer for terms that can serve as argu-
ment values for a domain relation or terms that serve as “trigger words” to identify a relation.

In the NFL-scoring domain, there is a closed class of NFLTeam that includes the only valid fillers for NFLTeam arguments. Valid fillers for the NFLGame argument are NPs that contain a word such as “victory,” “loss,” or “game.” We created recognizers for each argument type as well as two classes of trigger words: WinIndicator (victory, triumph, rout, and so on) and LossIndicator (loss, defeat, lose, and so on).

We implemented the class recognizers as regular expressions with class-specific lists of key words. This simple representation was to minimize the manual engineering effort and proved effective in practice. It required only a few minutes to identify the key words that were found in training examples and to supplement them with synonymous terms beyond those found in training. The recognizer for Date and for the NFL-scoring class FinalScore used special-purpose regular expressions.

In future work, we plan to develop automatic methods to find key words from training and then augment them using a combination of web search and WordNet synonyms. It is an open question how accurately this can be done with a small number of training examples and the inherent ambiguity of the key words (most NFLTeam instances in training are simply a city name).

Figure 5 shows one of the tuples from figure 1 after domain class recognizers have been applied to the tuple. The class recognizers have found that arg1 of the tuple contains a term that indicates winning a game, a phrase that denotes NFLGame, and two FinalScore instances. Similarly, arg2 contains an NFLTeam. In addition to labeling classes in the tuple arguments and predicate, we applied the class recognizers to the portion of the sentence to the left and to the right of the tuple.

The class recognizers have a tendency to overgeneralize: a city such as New Orleans is always labeled as the NFL team it hosts; scores such as the 28–0 halftime lead are incorrectly labeled as final scores. We found that this overgeneralization was not harmful in practice. We could learn high-precision domain mapping rules whose constraints compensated for the overgeneralized class labels.

Domain Relation Mapping Rules. The second step in domain adaptation takes tuples in which domain classes have been identified and maps these tuples to domain relations. Since this must be done with limited training, we chose an algorithm with strong learning biases — using a covering algorithm to learn rules that are expressed with constraints on classes and literals in specific arguments of the tuple.

Our rule representation has a set of constraints on tuple arguments and on context to the left or right of the tuple. If all constraints are met, the rule specifies a domain relation type, argument types, and the location of each argument value. A tuple argument may have multiple instances of a domain class, so a rule can specify that only the nth instance of that class is to be extracted. Finally, each rule has a confidence value.

We extended the rule learning to handle long-distance dependencies by including constraints on the “left context” (before arg1) and “right context” (after arg2). This enables extractions where the argument values are not found within a single tuple. Consider the sentence, “Kansas City took sole possession of the American Conference West lead with a 17–14 victory over Denver,” in which the relation of winning team and game is not a local reference. This was the case in about one-third of the training examples for GameWinner. A rule that applies to this sentence has constraints that an NFLTeam be in the left context and that WinIndicator, FinalScore, and NFLGame be found in arg1. This rule had seven correct and five errors on the training, for a confidence of 0.538.

Figure 6 shows two rules learned for the NFL-scoring domain. These rules apply successfully to our running example about a game between the New Orleans Saints and the San Francisco 49ers.

Our covering algorithm begins with a base rule for each seed instance in the training set, the most specific rule that covers the seed instance. Constraints in a base rule include all domain classes, literal constraints on all prepositions, and a position...
constraint that is set to 0 if the argument value is the first or only match for its class constraint (higher than 0 otherwise). This fairly restrictive rule representation was chosen to allow generalization from a small number of examples. A richer representation would require considerably more training.

The base rule, and each subsequent generalization of it, is applied to each tuple from the training sentences. If an extraction corresponds to a positive annotation, it is counted as correct, otherwise as an error. We compute rule confidence based on training precision with a variant of Laplacian smoothing: confidence = \( \frac{c}{c + e + 1} \), where there are \( c \) correct extractions and \( e \) errors. The rule learner keeps a beam set of the highest confidence \( k \) rules that cover the seed instance.

At each iteration, the rule learner generalizes each rule in the beam set by dropping a single constraint, considering each possible generalization. The new rules are added to the beam set and the learner continues until no further improvements are possible to the beam set. Constraints must not be dropped if they are specified as the subphrase to extract from a tuple argument. For example, the first rule in figure 6 cannot drop the constraint on NFLTeam in arg1 or the NFLGame constraint in arg2. Otherwise the rule cannot produce an extraction.

Active Learning for Domain Mapping

We also compensate for insufficient training annotations with active learning that presents a user with instances to tag as correct or error from a pool of unannotated tuples. Where traditional active learning selects instances to be classified, our active learning selects a rule \( r \) at each iteration. In our system, instances only arise from applying a rule to a tuple to produce extracted relations.

The system displays \( k \) random instances for the selected rule and their corresponding sentences, which a user tags as correct, error, or ignore. The learner then updates the number of correct and errors for \( r \) and for each specialization of \( r \) that applies to any of the newly tagged instances and adjusts the rule confidence.

The active learner starts with the set of all rules considered by the covering algorithm. Hence, it does not create any new rules, but adjusts the confidence in any rules that cover the seed instances from the original training set.

Our learner selects a rule to refine based on a combination of three metrics: confidence, saturation, and constraint penalty.

**Confidence** — prefer rules with higher precision on training. This keeps the learner focused on the most useful rules, those with high precision.

**Saturation** — apply a penalty to rules that already have adequate training: no penalty if the rule is undertrained (fewer than tagged instances); a maximum penalty of \( \frac{\beta - \alpha - 1}{\beta - \alpha} \) if there are \( \beta \) or more tagged instances; and a proportional penalty for between alpha and beta instances.

**Constraint penalty** — prefer rules with fewer constraint. Tagging instances for a general rule \( r \) will update the confidence in all specializations of \( r \) as well, so selecting general rules is more useful than specific rules. Apply a penalty of \( 1/\gamma \) for each constraint in the rule.

The metric for selecting a rule is
In these experiments, we set $\alpha = 10$, $\beta = 30$, and $\gamma = 20$. While not perfect, this selection policy tends to refine the highest-precision rules first and shifts to rules with less training after the high-precision rules have over 10 training examples.

**Empirical Results**

We evaluated TEXTRUNNER’s adaptation for the NFL domain on blind test sets of news articles from the English GigaWord corpus.¹ The test set was 100 articles that a classifier judged to be sports related. We first evaluated the output of rules learned from the initial annotations supplied by the DARPA Machine Reading Program. In a follow-up experiment we used active learning to improve results for each relation.

Once a rule set was learned, we ran TEXTRUNNER on each sentence in the test set and applied the concept recognizers and the relation mapping rules. When multiple rules extracted identical relations from the same sentence, we merged the extractions and took the highest rule confidence as the extraction confidence. We ranked extractions for each relation by confidence to create recall-precision curves. Recall is the number of correct extractions with confidence above a threshold, divided by the total number of correct extractions. Precision is the number correct divided by the number of extractions output above a confidence threshold.

Figure 7 shows recall and precision using rules learned from the initial annotation only. The results varied considerably between relations, depending not so much on the amount of training as on the locality of information. The relation TeamFinalScore, where both the NFLGame and the scores of the game were almost always found in the same tuple argument, had precision 0.95 at recall 0.87. The relation GameLoser, where the team name and game reference are usually found in the same or adjacent NPs (“victory over Denver,” “rout of Chicago,” and so on), had precision 0.93 at recall 0.32.

TEXTRUNNER had difficulty with the GameWinner relation due to nonlocal references. In about one third of the training examples the game winner was not found in the same tuple as the game reference. The rule learner needed to fall back on less reliable rules that looked for the team name in the “left context” of the sentence. This difficulty in

---

¹ Articles

**Figure 7. TEXTRUNNER Performance.**

Performance of TEXTRUNNER adaptation to the NFL-scoring domain on a blind test set of 100 NFL news articles. Even with limited training, TEXTRUNNER has precision above 0.90 for a significant fraction of the recall for three of these five relations.
identifying game winners also affected the performance of the TeamScoringAll relation, which must identify the score of both winning and losing teams.

Our experiment with active learning demonstrates that it can be effective in supplementing limited training data. Figure 8 shows results of active learning on the GameWinner relation. Even 20 new instances caused a dramatic improvement in the high-precision end of the curve, lifting it from precision 0.48 at recall 0.19 to precision 0.91 at recall 0.21. Precision increased with each new iteration of active learning, reaching 1.00 at recall 0.16 by 40 new instances and reaching precision 0.75 at recall 0.35 by 80 new instances. Tagging 80 instances took one of the authors about 25 minutes.

Active learning also gave similar boosts for the other relations from figure 7, lifting the recall-precision curve for each relation from a small number of new training examples. The relation TeamScoringAll required more new instances than GameWinner to show similar improvement, since TeamScoringAll needs examples of scores for both winning teams and losing teams.

Balancing Expressiveness and Learnability

We were able to learn high-precision rules from a small amount of training because of a restricted rule representation. TextRunner identifies instances of domain relations in a sentence with rules that only look at semantic tags and a small set of closed-class words. This enables the rules to generalize easily from a small number of examples. If it used arbitrary lexicalized features instead, or used sequences of words and parts of speech, the system would need thousands of training instances to gain traction.

The class recognizers for a domain provide type constraints on argument values, which is essential to maintaining high precision. Domain-specific classes of trigger words allow rules to generalize beyond the exact words found in training. The NFL-scoring domain had characteristics that fit this approach easily. Class recognizers could be created with only a few minutes of manual engineering for each of the argument types: NFLTeam, NFLGame, FinalScore, and Date.

Tagging phrases with these classes can be done independently of whether that class participates in a domain relation in a given sentence. Other domains that share this characteristic include business domains where person, corporation, corporate office, and monetary amount can be identified reliably. A biomedical domain will likewise have classes of medical terms that can be reliably tagged using a medical thesaurus.

The IC domain proved to be more challenging — unlike the NFL-scoring domain with its closed class of NFL...
teams, the agent and targets in the IC domain are open classes. A phrase cannot be tagged with a class such as Bombing/Agent, since any person, group of people, organization, or nation can be the agent of a bombing relation. This is a role that the phrase plays, rather than a class to which it belongs.

We experimented with augmenting class recognizers for the IC domain with generic named entity recognizers. This enables learning some high-precision rules, but has limited recall, because about half of the argument values are common nouns. Only a small fraction of the training annotations had a named entity for both arguments. The NER tags were also too coarse grained. A nation can be the agent of an attack, and a city can be a target, but the reverse never occurred in the annotations that implicitly define the relations. Yet, both nations and cities have the NER tag, Location.

We are exploring another strategy to provide the argument type checking that is needed for high precision, yet requires minimal manual engineering. In addition to the class constraints in particular rules, there will be argument type checking that is applied to all extractions for a domain relation. Argument phrases must match a disjunction of domain classes, NER tags, and domain-independent semantic classes such as those from WordNet. Thus fillers for the Agent argument of attack relations must have an NER tag Person or Organization or the head of the phrase must be a child of one of these WordNet senses: [person-1, organization-1, social group-1, country-1].

Related Work

Information extraction has a long history in the natural language-processing community going back to the Message Understanding Conferences² (Grishman and Sundheim 1996) and the Automated Content Extraction program.³ The first systems were rule based and highly domain dependent (Krupka et al. 1991, Hobbs et al. 1992, Riloff 1996, Miller et al. 1998). To achieve robustness under noisy settings rule-based systems were replaced by statistical and probabilistic methods using hidden Markov models, max-entropy models, and later conditional random fields (Skounakis, Craven, and Ray 2003; Lafferty, McCallum, and Pereira 2001; Culotta, McCallum, and Betz 2006). Hybrid systems attempted to combine the two paradigms (such as Califf and Mooney [1999]), but all these systems remained domain focused and required significant manual engineering and labeling effort.

A few systems, especially ones that focused on information extraction from the web, investigated open-domain IE (Etzioni et al. 2005; Talukdar et al. 2008; Moschitti, Morarescu, and Harabagiu 2003). Even though the methods were meant to be general and applied across different domains, still, the extractors needed to be relearned, often manually, for each relation of interest. With potentially thousands of relations of interest, these systems couldn't provide a convincing solution to the knowledge acquisition bottleneck. Our recent paradigm of Open IE (Banko et al. 2007, Banko and Etzioni 2008) overcomes these limitations by applying a self-learned extractor with unlexicalized features to separate out the relation expressed in the sentence as well as the entities between which the relation is expressed.

Reducing the cost of annotation effort is a well-studied problem in information extraction, active learning being one of many proposed solutions (Soderland 1999; Thompson, Califf, and Mooney 1999; Settles and Craven 2008). Other approaches include semi-supervised learning (Rosenfeld and Feldman 2007), metaboosrtstrapped learning (Riloff and Jones 1999), and more recently, transfer learning (Daume 2007).

Conclusions and Future Work

We have demonstrated that domain-independent extractions from an Open IE system can be mapped to a domain ontology with high precision from a small number of training examples. TEXTRUNNER with domain adaptation achieved precision over 0.90 for relations with as few as eight training examples from the NFL-scoring domain of the DARPA Machine Read-
Acknowledgments
This research was supported in part by NSF grant IIS0803481, ONR grant N00014-08-1-0431, and DARPA contract FA8750-09-C-0179, and carried out at the University of Washington’s Turing Center.

Notes
1. See ldc.upenn.edu.

References


Stephen Soderland is a research scientist at the Turing Center of the University of Washington. He received an MS and PhD in computer science from the University of Massachusetts in 1995 and 1997.

Brendan Roof is an undergraduate computer science and mathematics major at the University of Washington and a research assistant at the Turing Center.

Bo Qin received a BS and MS in computer science from the University of Washington in 2008 and 2010.

Shi (Will) Xu is an undergraduate computer science major at the University of Washington and a research assistant at the Turing Center.

Mausam is a research professor of computer science at the University of Washington. He received an MS and PhD in computer science at the University of Washington in 2004 and 2007.

Oren Etzioni is a professor of computer science at the University of Washington and the director of the Turing Center. He received a PhD in computer science from Carnegie Mellon University in 1991. He was elected Fellow of AAAI in 2003 for significant contributions to the fields of software agents, web-based technology, and intelligent user interfaces.