Extracting Action and Event Semantics from Web Text

Avirup Sil and **Fei Huang** and **Alexander Yates**
Temple University
Computer and Information Sciences
1805 N. Broad St.
Wachman Hall 324
Philadelphia, PA 19122
{avirup.sil,fei.huang,yates}@temple.edu

Abstract

Most information extraction research identifies the state of the world in text, including the entities and the relationships that exist between them. Much less attention has been paid to the understanding of dynamics, or how the state of the world changes over time. Because intelligent behavior seeks to change the state of the world in rational and utility-maximizing ways, common-sense knowledge about dynamics is essential for intelligent agents. In this paper, we describe a novel system, PREPOST, that tackles the problem of extracting the preconditions and effects of actions and events, two important kinds of knowledge for connecting world state and the actions that affect it. In experiments on Web text, PREPOST is able to improve by 79% over a baseline technique for identifying the effects of actions (64% improvement for preconditions).

1 Introduction

Very little information extraction research has investigated how to extract knowledge about *dynamics*, or how the state of the world changes over time. Most research focuses instead on understanding the state of the world and the objects in it. Relation extraction seeks to extract facts about entities in the world — a typical task is to identify where the headquarters of a company lies — and opinion mining seeks to identify people's beliefs about entities.

Yet common sense knowledge regarding the dynamics of the world is arguably the most crucial form of knowledge for an intelligent agent, since it informs an agent of the ways in which it can act upon the world. In other words, it provides the agent with the knowledge needed to *plan*. Artificial Intelligence (AI) researchers have recognized the central importance of planning in AI for a long time. Plan recognition (Schmidt, Sridharan, and Goodson 1978), or the task of inferring another agent's goals and plans based on limited observations of their behavior, has also been recognized as a central AI problem (Kautz 1991), as well as a fundamental problem in natural language understanding (Geib and Steedman 2007) and dialogue processing (Carberry 1990; Litman and Allen 1987).

Progress in these important areas, however, has always been limited to specialized domains, in part because of the knowledge acquisition bottleneck: it is time-consuming and labor-intensive to supply an AI system with the knowledge required to plan and recognize plans. Most AI planning, reasoning, and plan-recognition systems assume access to a well-organized representation of the domain in the form of actions and entities. While a great deal of work has gone into automatically extracting entities from text and placing them in an ontology, very little attention has been paid to the problem of extracting action and event semantics.

In this paper, we seek to expand the information extraction approach to knowledge acquisition so that it can acquire knowledge needed for important AI tasks. In particular, we seek to extract knowledge about dynamics: the preconditions and effects of actions and events. In order to overcome the knowledge acquisition bottleneck, we require a system that can scale to large and diverse text corpora like the Web, and that can extract knowledge for previously unseen actions and events with no manual effort.

We introduce the PREPOST system, which combines traditional text mining techniques with open-domain semantic role labeling to mine knowledge about action semantics. PREPOST uses a small set of labeled data to learn classifiers, but once it is trained, it is capable of identifying preconditions and postconditions for previously unseen actions and events with high precision and recall. In our experiments, we demonstrate that PREPOST can identify the preconditions of previously unseen actions using automatically downloaded Web documents with a precision of over 80% at 100% recall (80% precision at 77% recall for effects).

The next section provides formal background information on representing action semantics and describes previous work in related areas. Section 3 introduces the PREPOST system and details its techniques. Section 4 presents our experiments, and Section 5 concludes and outlines directions for future work.

2 Actions, Events, Preconditions, and Effects

2.1 Background and Terminology

We define *events* to be observable phenomena that happen at a particular time and place. Planning systems typically consider only certain kinds of events, called *actions*, which are events that are brought about by rational agents (as opposed to natural events and chance occurrences). Our extraction techniques work equally well on actions and non-agentive events. Events act upon *states* — conditions of the world and its objects and attributes — to change them over time.

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

put	
args:	x, y, z
pre:	agent(x), possesses(x, y), empty(z)
add:	on(y, z)
del:	possesses(x, y), empty(z)

Figure 1: An example STRIPS representation for the action `put`.

Because actions are central to AI, there is a long history of work in representing their semantics. One of the best-known, and still widely used, representations for action semantics is the STRIPS representation (Fikes and Nilsson 1971); an example of the STRIPS representation for the action `put` is given in Figure 1. Formally, a STRIPS representation is a 5-tuple $(a, args, pre, add, del)$ consisting of the action name a, a list $args$ of argument variables (sometimes with types), and three sets of states or conditions that reference the argument variables. The first, the *precondition* list pre, is a set of conditions that must be met in order for the action to be allowed to take place. For instance, in order for an agent to `put` a book on a table, the agent must possess the book. The other two sets of conditions specify how the world changes when the action takes place: the add list describes the set of new conditions that must be true afterwards (*e.g.*, `on(book,table)`), and the del list specifies the conditions that were true before the action happened but are no longer true. These add and del conditions are sometimes collectively referred to as *effects* or *postconditions*.

A full solution to the task of extracting STRIPS representations is beyond the scope of this paper, but we address two important subtasks. Using only a corpus C, we identify the set of words that describe either preconditions or postconditions to a. Furthermore, we classify which ones act as preconditions and which act as postconditions. A full solution would need to add to this with a technique for relating the extracted terms to particular arguments or sets of arguments; a technique for determining delete and add postconditions; and a technique for handling polysemous and synonymous extractions. We leave these tasks for future work.

2.2 Previous Work

Perhaps the most closely related recent work has investigated how to extract "narrative event schemas" (Chambers and Jurafsky 2009), or sets of events that often occur together. Schank and Abelson's (1977) famous example of a restaurant script includes events such as sitting down, ordering, eating, and paying the bill. Narrative event schemas often exhibit a natural temporal ordering; Chambers and Jurafsky (2008) have made progress on extracting this knowledge as well. The extracted schemas are distinct from the precondition and postcondition information that we seek to extract here in that they do not provide information about the world state. Our extracted knowledge could complement the restaurant script, for example, with knowledge that the diner is hungry before eating and full afterwards, and that as a result of paying the bill, the restaurant has earned money and the diner has lost it. This knowledge not only allows for a deeper understanding of the connections between events in scripts, but it also allows a system to reason about new connections and plan its own coherent sets of actions.

Other research into extracting the relationships between events has investigated causal relationships (Girju 2003; Chan et al. 2002; Khoo, Chan, and Niu 2000; Kaplan and Berry-Rogghe 1991; Garcia 1997) and, more generally, paraphrases, such as in the DIRT system (Lin and Pantel 2001). Such systems typically do not distinguish between event-event relationships — *e.g.*, rain causes flooding — and event-state relationships — *e.g.*, rain causes wet grass. Our system is focused only on the latter. Furthermore, these systems do not consider precondition relationships, which are neither causal nor paraphrases.

Numerous previous systems have investigated text mining techniques for common-sense knowledge acquisition. Two central aspects of these systems have been their focus on scalability and portability across domains (Banko and Etzioni 2008) and their ability to extract general-purpose knowledge from text (Schubert 2002). This allows their systems to extract as much knowledge as possible from large text repositories like the Web. We argue, however, that for many AI tasks, the kind of knowledge needed is special-purpose, like the preconditions and postconditions of actions. While domain portability and scalability are desirable, if we are to acquire knowledge for AI systems, we should focus on distinct extraction techniques for each of the (small finite number of) distinct AI tasks. Recent successes in this line of research include extracting knowledge from text for quantifier scope disambiguation (Srinivasan and Yates 2009) and for word sense disambiguation (Duan and Yates 2010), but as far as we are aware, no one has yet tackled extraction of action semantics.

3 Learning to Extract Action Semantics

We approach the problem of knowledge acquisition for actions and events by automatically extracting the knowledge from text. This approach has several advantages: most importantly, text regarding a huge number of different actions and events is readily and cheaply available from the Web and other document collections. This allows our common-sense knowledge acquisition process to scale to huge numbers of different actions in a wide variety of domains. Another advantage of our text mining approach is that as text mining systems process larger and larger collections of text, their accuracy tends to improve (*e.g.*, (Popescu and Etzioni 2005)). This allows the system to continuously improve and refine the extracted knowledge with minimal cost.

We have developed a text mining system, which we call PREPOST, for automatically collecting the preconditions and postconditions of actions and events from Web text. PREPOST involves two distinct learned classifiers, one for preconditions and one for postconditions, but the systems are identical in their architecture and differ only in the labels on the training data for learning. We now describe how the system harvests texts from the Web, determines candidate precondition and postcondition words, and learns a classifier for selecting the correct preconditions and postconditions from the candidates.

Collecting Text Given some action word A, PREPOST uses a search engine to find a large collection of documents that contain A. We search for the pattern "(is|are|was|were)

A-ing" to select documents for A. Conveniently, in English, this progressive form of the verb is used almost exclusively with event and action verbs, rather than stative verbs — "is buying" is perfectly acceptable because "buy" expresses an action, but "is owning" is awkward and uncommon because "own" expresses a state or condition. By selecting documents that contain the progressive form, we ensure that we select documents where the word A is being used as an action, rather than as a noun or perhaps a sense of the word A that is not an action. On average for the words in our experiments, PREPOST collects 426 documents (minimum of 160, maximum of 536). We refer to the document set for action A as D_A, and the entire document set as D.

Candidate Selection To narrow down the list of possible pre- and post-condition words, for every action word A, we compute the pointwise mutual information (PMI) between A and every word in D_A. In order to include possible deletion postconditions, we also include bigrams where the first word is "not" or "no." PMI is a well-known technique for measuring the degree of association between terms, and we calculate it as

$$\text{PMI}(A, w) = \log \frac{|\{d \in D_A | w \text{ appears in } d\}|}{|D_A| \times |\{d \in D | w \text{ appears in } d\}|}$$

We select the top 500 words w based on PMI scores as candidate pre- and post-conditions. On our manually labeled training data, the threshold of 500 was enough to ensure that over 95% of the correct pre- and post-conditions were among the candidates.

Basic Classifier Features Our aim is to build classifiers that are independent of the action words that are used as training examples. This requires that we use features that are not specific to individual action words, but still somehow capture the relationship between actions and preconditions, or between actions and postconditions.

The intuition behind the set of features for our classifiers is that we try to compute the degree of association and the nature of the association between every candidate word and action word. The PMI score is used to measure the degree of association, and is our first feature. Using this feature, a classifier can determine that "liquid" is a more likely postcondition for "melt" than is "gas", since "liquid" is much more highly correlated with "melt" in text and has a higher PMI.

We also want to allow the classifier to learn the nature of association between candidate words and action words. "Ice" is highly correlated with "melt", but is neither a precondition nor postcondition; it is simply a common argument type. In order to distinguish words like "ice" from true preconditions like "solid," we aim to determine the set of words that characterize the relationship between the candidate word and the action word. For instance, knowing that "melt", "heat", and "requires" have a strong 3-way association is an indicator that "heat" is not just related to "melt", but is actually a precondition. With the right set of third-party words like "requires" and a measure of 3-way association between terms, we can characterize how action words and candidate words relate to one another.

We build a vector space model in which each dimension is a "feature word", or word that may characterize how the action word and candidate word are related. For a given action word A and candidate word C, we then measure the three-way PMI between A, C, and each feature word F:

$$\text{PMI} = \log \frac{|\{d \in D_A | C, F \in d\}|}{|D_A||\{d \in D | w \in d\}||\{d \in D | F \in d\}|}$$

The resulting feature vector for A and C is a characterization of the vocabulary that correlates highly with A and C, and can be very useful for distinguishing between precondition words, which should have a high PMI with words like "required" and "before", and non-precondition candidate words that happen to have high PMI with A.

As a feature selection technique to determine a set of feature words we used χ^2 which worked quite well. We measured the frequency of all words in our entire document collection, and sorted. We then selected all of words that occurred above a threshold T number of times, to ensure that our PMI statistics would have sufficient numbers of occurrences to work with. Using $T = 500$, we ended up with around 3000 candidate feature words. We then computed χ^2 values by comparing each candidate feature word and our labeled pre and postconditions. Choosing an experimental threshold on the value of χ^2 we came up with 161 feature words for doing our experiments to extract both pre and postconditions.

to remove numbering (before each equation)

Features from extracted relationships: Thus far our feature set has considered only coarse-grained, document-level features that ignore the significant structure in language and the prediction task. We next consider ways to annotate the text with structural information that is useful for identifying action semantics. As an example, consider the sentence

> "Kava is a highly prized medicinal herb whose primary benefit is alleviating anxiety and minor **pain**."

For many actions, like "alleviate," the arguments to the action in a sentence often refer to preconditions and postconditions, like the precondition "pain". A logical expression for the above sentence would include expressions like the following:

`alleviate(`x, y`)` $\wedge$ `kava(`x`)` $\wedge$ `pain(`y`)`

By annotating our texts to indicate predicates like `alleviate` and arguments like `pain`, we can measure how often candidate words are associated through predicate-argument structure with certain actions.

We annotate our texts using a recently-developed open-domain semantic role labeling (SRL) system (Huang and Yates 2010). SRL identifies predicates (usually verbs) in text, finds the phrases that constitute the arguments to the predicates, and classifies the arguments into roles, including the agent of the action, the beneficiary, the thing being acted upon, the location, and a host of other possible role types. Huang and Yates's system has state-of-the-art performance on text taken from domains other than newswire text, as is the case for Web text. We also apply coreference resolution using OpenNLP's maximum-entropy coreference classifier[1].

[1] http://opennlp.sourceforge.net/

Using this annotated text, we compute the following features: how often the candidate word C appears as part of a phrase that is an argument to the action word A, and how often the candidate word appears as an argument to any predicate at all. This feature helps provide less sparse feature counts than the first feature.

To help provide less sparse counts that still connect the candidate word and the action word, we use a distance-based feature and two coreference-based features. Our distance-based feature counts how often C appears as an argument near (less than 50 words away from) A. The two coreference-based features count how often C appears as an argument that is coreferential with an argument of A, and how often C appears as an argument to a predicate P which has another argument D which is coreferential with an argument of A. This last feature is useful for sentence pairs like the following from our texts:

Doctors need to heal patients$_1$.
They$_1$ are the ones who need **medicine**.

Here, "medicine" is not directly connected to "heal", but the two words are associated indirectly through coreference between "They" and "patients".

SVM Learning: We use support vector machines (SVMs) as our learning machines, implemented in SVM-Light (Joachims 1999). We use a radial basis function kernel with default parameter settings.

4 Experiments

In our experiments, we test the ability of our extraction technique to find preconditions and postconditions for actions it has never seen before.

4.1 Experimental Setup

We randomly selected a set of 40 actions from the lexical units in the frames that inherit from the `Transitive_action` frame in FrameNet (Johnson et al. 2003). For each action word, we hand-constructed a STRIPS-like representation of the action. As described above, we automatically downloaded Web documents for each action word, and constructed a list of candidate pre- and post-conditions for each. Candidate action words that matched our hand-constructed STRIPS representations were marked as positive examples; otherwise, we treated them as negative examples. On average, our hand-labeled data had 4.2 preconditions (minimum of 3, maximum of 6) and 3 postconditions (minimum of 2, maximum of 6). We used five action words as training data (alleviate, arrange, awaken, behead, boil) and thirty-five action words as test data (cut, humidify, defrost, desiccate, harden, heal, magnify etc).

We evaluate our extraction system based on precision and recall. We test our learned model by ranking the candidate words for an action by the SVM's predicted value of $\vec{w}^T\vec{x} + b$. We then vary a threshold t between 1 and the number of candidate words, and determine a precision-recall curve where the precision and recall at t are given by:

$$\text{precision}_t(aw) = \frac{|\{cw|\text{precond}(aw, cw), \text{rank}(cw) < t\}|}{|\{cw|\text{rank}(cw) < t\}|}$$

$$\text{recall}_t(aw) = \frac{|\{cw|\text{precond}(aw, cw), \text{rank}(cw) < t\}|}{|\{cw|\text{precond}(aw, cw)\}|}$$

where $\text{precond}(aw, cw)$ is true when cw is a precondition of aw in our labeled data, and $\text{rank}(cw) < t$ is true if cw is among the top $t-1$ candidate words in our ranked list. The same procedure is applied for postconditions. As a baseline for our experiments, we rank according to the PMI between a candidate word and an action word.

4.2 Results and Discussion

The PREPOST system achieves remarkable success on this highly skewed prediction task. Figure 2 shows the precision-recall curves for several variations of our learned model on detecting preconditions; Figure 3 shows our postcondition results. We also found out that the AUCs for all the curves corresponding to the variations of our model were statistically significantly different with *p*-value much less than 0.001. We tested each variation of our model using un-paired *t*-test. The PREPOST system improves over the baseline PMI ranking by over 0.57 in AUC for both preconditions and postconditions. It improves over the SVM model that combines PMI and feature words by over 0.39 in AUC for both preconditions and postconditions. It achieves a 94.3% precision at 85.4% recall on preconditions (93.5% for postconditions).

We also did an experiment using fractions of our Web text, and we observed that the performance of our system kept on increasing with the increase in Web text. Figures 4 and 5 show our results. These promising results suggest that there is room for further improvement by adding in larger quantities of text to our corpora.

To determine which sets of features were most important to the success of PREPOST, we ran several variations of our experiments in which we deleted certain features from the complete model. Deleting either our two coreference-based features or the distance-based feature (PREPOST-Coref and PREPOST-Dist) caused the system to drop 0.1 to 0.15 points in AUC for preconditions and 0.3 to 0.35 points for postconditions. Deleting the coreference features causes slightly larger drops for both preconditions and postconditions, suggesting that they are more valuable to the model. The coreference-based features are sparser than the distance feature, but they apparently make up for this with the more semantically-meaningful connection between the candidate word and action word. Including both sets of features causes a large jump in performance, suggesting that the two features are not redundant.

Interestingly, when we deleted both coreference and distance-based features from the model, the resulting classifier performs worse than the PMI + feature words classifier that has no semantic role labeling features. We attribute this to two causes: first, the feature which counts how often the candidate word is an argument to the action word is too sparse to be useful. We need to use either coreference or distance thresholds to connect the action word with precondition and postcondition words that are farther from the action word. Second, the feature which counts how often the candidate word appears as an argument to any predicate contributes too much noise; too many incorrect candidate words appear as part of some argument.

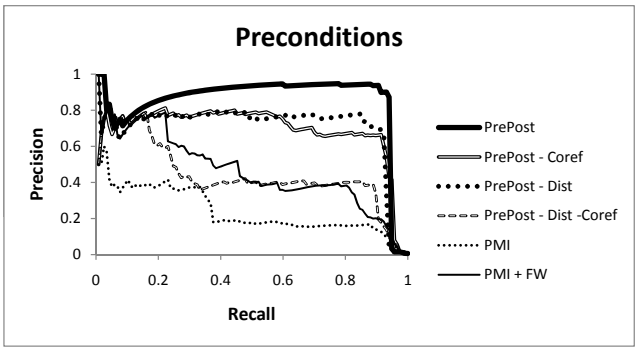

Figure 2: Precision-recall plots for extracting preconditions using variations of our models. "–" indicates the feature has been removed from the PREPOST model.

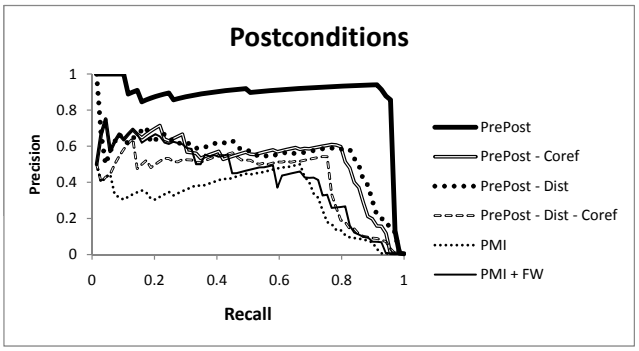

Figure 3: Precision-recall plots for extracting postconditions using variations of our models. "–" indicates the feature has been removed from the PREPOST model.

Table 1 shows the five candidate words that PREPOST ranks highest for preconditions and postconditions of the action "cut." Most of the errors that PREPOST makes are common arguments to the action word, like "object" and "doctor" for "cut," or "body" and "touch" for "heal." For the action word "defrost," it ranks "food" highly as a precondition. Certain prepositions, like "against" and "after," commonly appear as parts of arguments to our action words, and get ranked highly by PREPOST; these could easily be eliminated with stopword lists. In general, these mistakes show that PREPOST places significant weight on the semantic role labeling features, but it needs to better distinguish between common argument words and words that indicate conditions of the world state before and after the action takes place. Better selection of feature words might help, as well as using the role type information from semantic role labeling in the features.

Another common mistake is that PREPOST at times confuses preconditions and postconditions – for example, it ranks "relief" highly for preconditions of "heal", and "hard" ranks highly as a precondition for the action "harden." In other cases, PREPOST extracted terms which are not labeled as correct, but which in hindsight could have been part of our hand-labeled STRIPS representation (*e.g.*, "wound" as a precondition for "heal").

Word sense ambiguity was not as big of a problem with

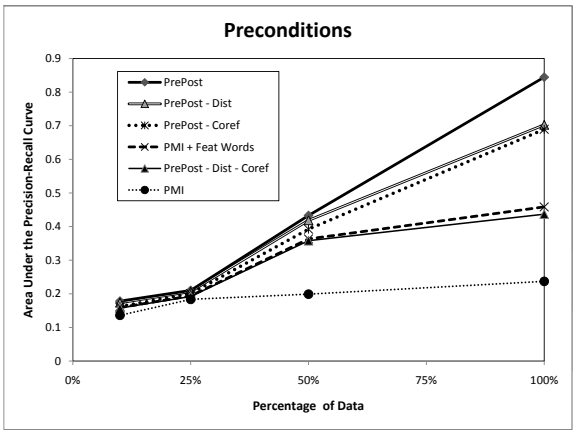

Figure 4: Performance of our model variations for extracting preconditions with gradual increment of Web Text.

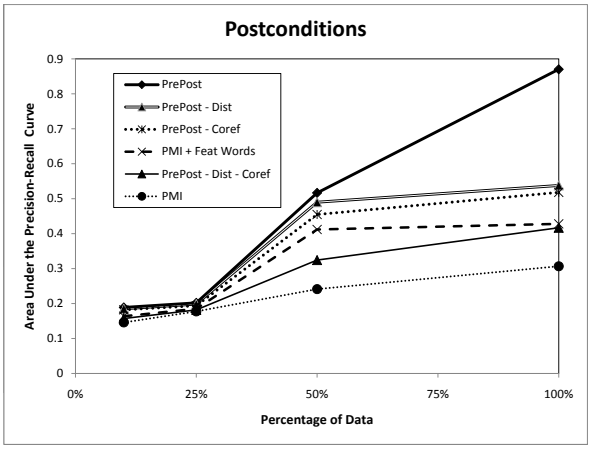

Figure 5: Performance of our model variations for extracting postconditions with gradual increment of Web Text.

our extraction process as we expected at first. This may be in part because of the particular words in our test set, which appear to have a single dominant sense in our document collection. More experimentation is needed to investigate this issue.

5 Conclusion and Future Work

In a first attempt at acquiring knowledge of how the world changes over time, we have demonstrated that the PREPOST system is capable of extracting names for the precondition and postcondition relationships of actions with high accuracy. Using unlabeled documents from the Web, the trained PREPOST system can identify the small subset of the words in this text that reflect how an action word affects world state. While more experiments are clearly needed, our early results point to the exciting possibility that we may be able to acquire the knowledge that allows AI systems to plan and understand dialogues in a domain-independent manner.

There is still significant room to improve the accuracy of our existing extraction mechanism, perhaps by using fea-

Precondition		Postcondition	
$w^T x + b$	term	$w^T x + b$	term
4.23	**knife**	4.84	**blood**
3.82	**sharp**	0.95	**bloodshed**
1.89	object	-0.75	knife
1.84	**person**	-0.83	advanced
-0.71	body	-0.83	doctor

Table 1: Top 5 extracted preconditions and postconditions for the action word "cut." Extractions labeled correct are marked in bold.

ture selection tools to identify phrases and lexical patterns that help distinguish preconditions and postconditions. Beyond that, we need techniques for organizing the extractions into a coherent STRIPS representation. This will include breaking down the postconditions into add and delete effects; identifying logical connectives between the extracted conditions; and identifying which arguments of the action words are constrained by the preconditions and postconditions. Moreover, many recent planning techniques have had to go beyond the STRIPS representation to handle such phenomena as durative actions, actions with stochastic effects, and actions with continuous arguments. There is a great deal of room for enhancing the extracted knowledge in PREPOST to yield more sophisticated representations of actions. While the long-term benefits of this knowledge acquisition process remain distant, there are also language processing applications in the nearer future. As one example, a system could use knowledge of preconditions and postconditions to supplement explicitly mentioned actions and effects in a document with the implicit information about the world state before and after the actions, potentially improving recall in relation extraction.

References

Banko, M., and Etzioni, O. 2008. The tradeoffs between open and traditional information extraction. In *Proceedings of the ACL*.

Carberry, S. 1990. *Plan Recognition in Natural Language Dialogue*. Cambridge, MA, USA: MIT Press.

Chambers, N., and Jurafsky, D. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP*, 698–706.

Chambers, N., and Jurafsky, D. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL-IJCNLP 2009*.

Chan, K.; Low, B.-T.; Lam, W.; and Lam, K.-P. 2002. Extracting causation knowledge from natural language texts. In *Advances in Knowledge Discovery and Data Mining*. Springer. 555–560.

Duan, W., and Yates, A. 2010. Extracting glosses to disambiguate word senses. In *Proc. of HLT-NAACL*.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3/4):189–208.

Garcia, D. 1997. COATIS, an NLP system to locate expressions of actions connected by causality links. In *Proceedings of the 10th European Workshop on Knowledge Acquisition, Modeling and Management*.

Geib, C. W., and Steedman, M. 2007. On natural language processing and plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1612–1617.

Girju, R. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering*, 76–83. Morristown, NJ, USA: Association for Computational Linguistics.

Huang, F., and Yates, A. 2010. Open-domain semantic role labeling by modeling word spans. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Joachims, T. 1999. Making large-scale SVM learning practical. In Schölkopf, B.; Burges, C.; and Smola, A., eds., *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

Johnson, C.; Petruck, M.; Baker, C.; Ellsworth, M.; Ruppenhofer, J.; and Fillmore, C. 2003. *FrameNet: Theory and practice*.

Kaplan, R. M., and Berry-Rogghe, G. 1991. Knowledge based acquisition of causal relationships in text. *Knowledge Acquisition* 3:317–337.

Kautz, H. A. 1991. A formal theory of plan recognition and its implementation. In *Reasoning about plans*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 69–124.

Khoo, C. S. G.; Chan, S.; and Niu, Y. 2000. Extracting causal knowledge from a medical database using graphical patterns. In *Proceedings of ACL*, 336–343. Morristown, NJ, USA: Association for Computational Linguistics.

Lin, D., and Pantel, P. 2001. DIRT – Discovery of Inference Rules from Text. In *KDD*.

Litman, D. J., and Allen, J. F. 1987. A plan recognition model for subdialogues in conversations. *Cognitive Science* 11(2):163 – 200.

Popescu, A., and Etzioni, O. 2005. Extracting product features and opinions from reviews. In *EMNLP*.

Schank, R., and Abelson, R. 1977. *Scripts, plans, goals and understanding: an inquiry into human knowledge structures*. Erlbaum.

Schmidt, C.; Sridharan, N.; and Goodson, J. 1978. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence* 11(1,2).

Schubert, L. 2002. Can we derive general world knowledge from texts. In *Procs. of Human Language Technology Conference*.

Srinivasan, P., and Yates, A. 2009. Quantifier scope disambiguation using extracted pragmatic knowledge: Preliminary results. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.