Learning High-Level Planning from Text

S.R.K. Branavan, Nate Kushman, Tao Lei, Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{branavan, nkushman, taolei, regina}@csail.mit.edu

Abstract

Comprehending action preconditions and effects is an essential step in modeling the dynamics of the world. In this paper, we express the semantics of precondition relations extracted from text in terms of planning operations. The challenge of modeling this connection is to ground language at the level of relations. This type of grounding enables us to create high-level plans based on language abstractions. Our model jointly learns to predict precondition relations from text and to perform high-level planning guided by those relations. We implement this idea in the reinforcement learning framework using feedback automatically obtained from plan execution attempts. When applied to a complex virtual world and text describing that world, our relation extraction technique performs on par with a supervised baseline, yielding an F-measure of 66% compared to the baseline's 65%. Additionally, we show that a high-level planner utilizing these extracted relations significantly outperforms a strong, text unaware baseline – successfully completing 80% of planning tasks as compared to 69% for the baseline.[1]

A pickaxe, which is used to harvest stone, can be made from wood.

(a)

Low Level Actions for: wood → pickaxe → stone

step 1:	move from (0,0) to (2,0)
step 2:	chop tree at: (2,0)
step 3:	get wood at: (2,0)
step 4:	craft plank from wood
step 5:	craft stick from plank
step 6:	craft pickaxe from plank and stick
· · ·	
step N-1:	pickup tool: pickaxe
step N:	harvest stone with pickaxe at: (5,5)

(b)

Figure 1: Text description of preconditions and effects (a), and the low-level actions connecting them (b).

1 Introduction

Understanding action preconditions and effects is a basic step in modeling the dynamics of the world. For example, having *seeds* is a precondition for growing *wheat*. Not surprisingly, preconditions have been extensively explored in various sub-fields of AI. However, existing work on action models has largely focused on tasks and techniques specific to individual sub-fields with little or no interconnection between them. In NLP, precondition relations have been studied in terms of the linguistic mechanisms that realize them, while in classical planning, these relations are viewed as a part of world dynamics. In this paper, we bring these two parallel views together, grounding the linguistic realization of these relations in the semantics of planning operations.

The challenge and opportunity of this fusion comes from the mismatch between the abstractions of human language and the granularity of planning primitives. Consider, for example, text describing a virtual world such as Minecraft[2] and a formal description of that world using planning primitives. Due to the mismatch in granularity, even the simple relations between *wood*, *pickaxe* and *stone* described in the sentence in Figure 1a results in dozens of low-level planning actions in the world, as can be seen in Figure 1b. While the text provides a high-level description of world dynamics, it does not provide sufficient details for successful plan execution. On the other hand, planning with low-level actions does not suffer from this limitation, but is computationally intractable for even moderately complex tasks. As a consequence, in many practical domains, planning algorithms rely on manually-crafted high-level

[1]The code, data and experimental setup for this work are available at http://groups.csail.mit.edu/rbg/code/planning

[2]http://www.minecraft.net/

abstractions to make search tractable (Ghallab et al., 2004; Lekavý and Návrat, 2007).

The central idea of our work is to express the semantics of precondition relations extracted from text in terms of planning operations. For instance, the precondition relation between pickaxe and stone described in the sentence in Figure 1a indicates that plans which involve obtaining stone will likely need to first obtain a pickaxe. The novel challenge of this view is to model grounding at the level of relations, in contrast to prior work which focused on object-level grounding. We build on the intuition that the validity of precondition relations extracted from text can be informed by the execution of a low-level planner.[3] This feedback can enable us to learn these relations without annotations. Moreover, we can use the learned relations to guide a high level planner and ultimately improve planning performance.

We implement these ideas in the reinforcement learning framework, wherein our model jointly learns to predict precondition relations from text and to perform high-level planning guided by those relations. For a given planning task and a set of candidate relations, our model repeatedly predicts a sequence of subgoals where each subgoal specifies an attribute of the world that must be made true. It then asks the low-level planner to find a plan between each consecutive pair of subgoals in the sequence. The observed feedback – whether the low-level planner succeeded or failed at each step – is utilized to update the policy for both text analysis and high-level planning.

We evaluate our algorithm in the Minecraft virtual world, using a large collection of user-generated online documents as our source of textual information. Our results demonstrate the strength of our relation extraction technique – while using planning feedback as its only source of supervision, it achieves a precondition relation extraction accuracy on par with that of a supervised SVM baseline. Specifically, it yields an F-score of 66% compared to the 65% of the baseline. In addition, we show that these extracted relations can be used to improve the performance of a high-level planner. As baselines for this evaluation, we employ the Metric-FF planner (Hoffmann and Nebel, 2001),[4] as well as a text-unaware variant of our model. Our results show that our text-driven high-level planner significantly outperforms all baselines in terms of completed planning tasks – it successfully solves 80% as compared to 41% for the Metric-FF planner and 69% for the text unaware variant of our model. In fact, the performance of our method approaches that of an oracle planner which uses manually-annotated preconditions.

2 Related Work

Extracting Event Semantics from Text The task of extracting preconditions and effects has previously been addressed in the context of lexical semantics (Sil et al., 2010; Sil and Yates, 2011). These approaches combine large-scale distributional techniques with supervised learning to identify desired semantic relations in text. Such combined approaches have also been shown to be effective for identifying other relationships between events, such as causality (Girju and Moldovan, 2002; Chang and Choi, 2006; Blanco et al., 2008; Beamer and Girju, 2009; Do et al., 2011).

Similar to these methods, our algorithm capitalizes on surface linguistic cues to learn preconditions from text. However, our only source of supervision is the feedback provided by the planning task which utilizes the predictions. Additionally, we not only identify these relations in text, but also show they are valuable in performing an external task.

Learning Semantics via Language Grounding Our work fits into the broad area of grounded language acquisition, where the goal is to learn linguistic analysis from a situated context (Oates, 2001; Siskind, 2001; Yu and Ballard, 2004; Fleischman and Roy, 2005; Mooney, 2008a; Mooney, 2008b; Branavan et al., 2009; Liang et al., 2009; Vogel and Jurafsky, 2010). Within this line of work, we are most closely related to the reinforcement learning approaches that learn language by interacting with an external environment (Branavan et al., 2009; Branavan et al., 2010; Vogel and Jurafsky, 2010; Branavan et al., 2011).

[3] If a planner can find a plan to successfully obtain stone after obtaining a pickaxe, then a pickaxe is likely a precondition for stone. Conversely, if a planner obtains stone without first obtaining a pickaxe, then it is likely not a precondition.

[4] The state-of-the-art baseline used in the 2008 International Planning Competition. http://ipc.informatik.uni-freiburg.de/

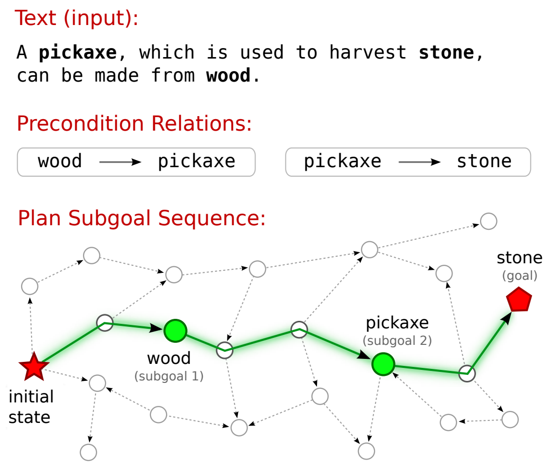

Figure 2: A high-level plan showing two subgoals in a precondition relation. The corresponding sentence is shown above.

The key distinction of our work is the use of grounding to learn abstract pragmatic relations, i.e. to learn linguistic patterns that describe relationships between objects in the world. This supplements previous work which grounds words to objects in the world (Branavan et al., 2009; Vogel and Jurafsky, 2010). Another important difference of our setup is the way the textual information is utilized in the situated context. Instead of getting step-by-step instructions from the text, our model uses text that describes general knowledge about the domain structure. From this text, it extracts relations between objects in the world which hold independently of any given task. Task-specific solutions are then constructed by a planner that relies on these relations to perform effective high-level planning.

Hierarchical Planning It is widely accepted that high-level plans that factorize a planning problem can greatly reduce the corresponding search space (Newell et al., 1959; Bacchus and Yang, 1994). Previous work in planning has studied the theoretical properties of valid abstractions and proposed a number of techniques for generating them (Jonsson and Barto, 2005; Wolfe and Barto, 2005; Mehta et al., 2008; Barry et al., 2011). In general, these techniques use static analysis of the low-level domain to induce effective high-level abstractions. In contrast, our focus is on learning the abstraction from natural language. Thus our technique is complementary to past work, and can benefit from human knowledge about the domain structure.

3 Problem Formulation

Our task is two-fold. First, given a text document describing an environment, we wish to extract a set of precondition/effect relations implied by the text. Second, we wish to use these induced relations to determine an action sequence for completing a given task in the environment.

We formalize our task as illustrated in Figure 2. As input, we are given a world defined by the tuple $\langle S, A, T \rangle$, where S is the set of possible world states, A is the set of possible actions and T is a deterministic state transition function. Executing action a in state s causes a transition to a new state s' according to $T(s' \mid s, a)$. States are represented using propositional logic predicates $x_i \in X$, where each state is simply a set of such predicates, i.e. $s \subset X$.

The objective of the text analysis part of our task is to automatically extract a set of valid precondition/effect relationships from a given document d. Given our definition of the world state, preconditions and effects are merely single term predicates, x_i, in this world state. We assume that we are given a seed mapping between a predicate x_i, and the word types in the document that reference it (see Table 3 for examples). Thus, for each predicate pair $\langle x_k, x_l \rangle$, we want to utilize the text to predict whether x_k is a precondition for x_l; i.e., $x_k \rightarrow x_l$. For example, from the text in Figure 2, we want to predict that possessing a pickaxe is a precondition for possessing stone. Note that this relation implies the reverse as well, i.e. x_l can be interpreted as the effect of an action sequence performed on state x_k.

Each planning goal $g \in G$ is defined by a starting state s_0^g, and a final goal state s_f^g. This goal state is represented by a set of predicates which need to be made true. In the planning part of our task our objective is to find a sequence of actions $\vec{a}$ that connect s_0^g to s_f^g. Finally, we assume document d does not contain step-by-step instructions for any individual task, but instead describes general facts about the given world that are useful for a wide variety of tasks.

4 Model

The key idea behind our model is to leverage textual descriptions of preconditions and effects to guide the construction of high level plans. We define a high-level plan as a sequence of *subgoals*, where each subgoal is represented by a single-term predicate, x_i, that needs to be set in the corresponding world state – e.g. `have(wheat)=true`. Thus the set of possible subgoals is defined by the set of all possible single-term predicates in the domain. In contrast to low-level plans, the transition between these subgoals can involve multiple low-level actions. Our algorithm for textually informed high-level planning operates in four steps:

1. Use text to predict the preconditions of each subgoal. These predictions are for the entire domain and are not goal specific.
2. Given a planning goal and the induced preconditions, predict a subgoal sequence that achieves the given goal.
3. Execute the predicted sequence by giving each pair of consecutive subgoals to a low-level planner. This planner, treated as a black-box, computes the low-level plan actions necessary to transition from one subgoal to the next.
4. Update the model parameters, using the low-level planner's success or failure as the source of supervision.

We formally define these steps below.

Modeling Precondition Relations Given a document d, and a set of subgoal pairs $\langle x_i, x_j \rangle$, we want to predict whether subgoal x_i is a precondition for x_j. We assume that precondition relations are generally described within single sentences. We first use our seed grounding in a preprocessing step where we extract all predicate pairs where both predicates are mentioned in the same sentence. We call this set the *Candidate Relations*. Note that this set will contain many invalid relations since co-occurrence in a sentence does not necessarily imply a valid precondition relation.[5] Thus for each sentence, $\vec{w}_k$, associated with a given *Candidate Relation*, $x_i \rightarrow x_j$, our task is to predict whether the sentence indicates the relation. We model this decision via a log linear distribution as follows:

$$p(x_i \rightarrow x_j \mid \vec{w}_k, q_k; \theta_c) \propto e^{\theta_c \cdot \phi_c(x_i, x_j, \vec{w}_k, q_k)}, \quad (1)$$

where θ_c is the vector of model parameters. We compute the feature function ϕ_c using the seed grounding, the sentence $\vec{w}_k$, and a given dependency parse q_k of the sentence. Given these per-sentence decisions, we predict the set of all valid precondition relations, C, in a deterministic fashion. We do this by considering a precondition $x_i \rightarrow x_j$ as valid if it is predicted to be valid by at least one sentence.

[5]In our dataset only 11% of *Candidate Relations* are valid.

```
Input: A document $d$, Set of planning tasks $G$,
       Set of candidate precondition relations $C_{all}$,
       Reward function $r()$, Number of iterations $T$
Initialization: Model parameters $\theta_x = 0$ and $\theta_c = 0$.
for $i = 1 \cdots T$ do
    Sample valid preconditions:
    $C \leftarrow \emptyset$
    foreach $\langle x_i, x_j \rangle \in C_{all}$ do
        foreach Sentence $\vec{w}_k$ containing $x_i$ and $x_j$ do
            $v \sim p(x_i \rightarrow x_j \mid \vec{w}_k, q_k; \theta_c)$
            if $v = 1$ then $C = C \cup \langle x_i, x_j \rangle$
        end
    end
    Predict subgoal sequences for each task $g$.
    foreach $g \in G$ do
        Sample subgoal sequence $\vec{x}$ as follows:
        for $t = 1 \cdots n$ do
            Sample next subgoal:
            $x_t \sim p(x \mid x_{t-1}, s_0^g, s_f^g, C; \theta_x)$
            Construct low-level subtask from $x_{t-1}$ to $x_t$
            Execute low-level planner on subtask
        end
        Update subgoal prediction model using Eqn. 2
    end
    Update text precondition model using Eqn. 3
end
```

Algorithm 1: A policy gradient algorithm for parameter estimation in our model.

Modeling Subgoal Sequences Given a planning goal g, defined by initial and final goal states s_0^g and s_f^g, our task is to predict a sequence of subgoals $\vec{x}$ which will achieve the goal. We condition this decision on our predicted set of valid preconditions C, by modeling the distribution over sequences $\vec{x}$ as:

$$p(\vec{x} \mid s_0^g, s_f^g, C; \theta_x) = \prod_{t=1}^{n} p(x_t \mid x_{t-1}, s_0^g, s_f^g, C; \theta_x),$$

$$p(x_t \mid x_{t-1}, s_0^g, s_f^g, C; \theta_x) \propto e^{\theta_x \cdot \phi_x(x_t, x_{t-1}, s_0^g, s_f^g, C)}.$$

Here we assume that subgoal sequences are Markovian in nature and model individual subgoal predictions using a log-linear model. Note that in contrast to Equation 1 where the predictions are goal-agnostic, these predictions are goal-specific. As before, θ_x is the vector of model parameters, and ϕ_x is the feature function. Additionally, we assume a special stop symbol, $x_\emptyset$, which indicates the end of the subgoal sequence.

Parameter Update Parameter updates in our model are done via reinforcement learning. Specifically, once the model has predicted a subgoal sequence for a given goal, the sequence is given to the low-level planner for execution. The success or failure of this execution is used to compute the reward signal r for parameter estimation. This predict-execute-update cycle is repeated until convergence. We assume that our reward signal r strongly correlates with the correctness of model predictions. Therefore, during learning, we need to find the model parameters that maximize expected future reward (Sutton and Barto, 1998). We perform this maximization via stochastic gradient ascent, using the standard policy gradient algorithm (Williams, 1992; Sutton et al., 2000).

We perform two separate policy gradient updates, one for each model component. The objective of the text component of our model is purely to predict the validity of preconditions. Therefore, subgoal pairs $\langle x_k, x_l \rangle$, where x_l is reachable from x_k, are given positive reward. The corresponding parameter update, with learning rate α_c, takes the following form:

$$\Delta\theta_c \leftarrow \alpha_c\, r \Big[\phi_c(x_i, x_j, \vec{w}_k, q_k) - \mathbb{E}_{p(x_{i'} \rightarrow x_{j'}|\cdot)}\left[\phi_c(x_{i'}, x_{j'}, \vec{w}_k, q_k)\right]\Big]. \quad (2)$$

The objective of the planning component of our model is to predict subgoal sequences that successfully achieve the given planning goals. Thus we directly use plan-success as a binary reward signal, which is applied to each subgoal decision in a sequence. This results in the following update:

$$\Delta\theta_x \leftarrow \alpha_x\, r \sum_t \Big[\phi_x(x_t, x_{t-1}, s_0^g, s_f^g, C) - \mathbb{E}_{p(x'_t|\cdot)}\left[\phi_x(x'_t, x_{t-1}, s_0^g, s_f^g, C)\right]\Big], \quad (3)$$

where t indexes into the subgoal sequence and α_x is the learning rate.

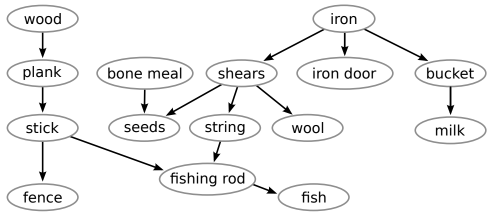

Figure 3: Example of the precondition dependencies present in the Minecraft domain.

Domain	#Objects	#Pred Types	#Actions
Parking	49	5	4
Floortile	61	10	7
Barman	40	15	12
Minecraft	**108**	**16**	**68**

Table 1: A comparison of complexity between Minecraft and some domains used in the IPC-2011 sequential satisficing track. In the Minecraft domain, the number of objects, predicate types, and actions is significantly larger.

5 Applying the Model

We apply our method to Minecraft, a grid-based virtual world. Each grid location represents a tile of either land or water and may also contain resources. Users can freely move around the world, harvest resources and craft various tools and objects from these resources. The dynamics of the world require certain resources or tools as prerequisites for performing a given action, as can be seen in Figure 3. For example, a user must first craft a *bucket* before they can collect *milk*.

Defining the Domain In order to execute a traditional planner on the Minecraft domain, we define the domain using the Planning Domain Definition Language (PDDL) (Fox and Long, 2003). This is the standard task definition language used in the International Planning Competitions (IPC).[6] We define as predicates all aspects of the game state – for example, the location of resources in the world, the resources and objects possessed by the player, and the player's location. Our subgoals x_i and our task goals s_f^g map directly to these predicates. This results in a domain with significantly greater complexity than those solvable by traditional low-level planners. Table 1 compares the complexity of our domain with some typical planning domains used in the IPC.

[6]http://ipc.icaps-conference.org/

Low-level Planner As our low-level planner we employ Metric-FF (Hoffmann and Nebel, 2001), the state-of-the-art baseline used in the 2008 International Planning Competition. Metric-FF is a forward-chaining heuristic state space planner. Its main heuristic is to simplify the task by ignoring operator delete lists. The number of actions in the solution for this simplified task is then used as the goal distance estimate for various search strategies.

Features The two components of our model leverage different types of information, and as a result, they each use distinct sets of features. The text component features ϕ_c are computed over sentences and their dependency parses. The Stanford parser (de Marneffe et al., 2006) was used to generate the dependency parse information for each sentence. Examples of these features appear in Table 2. The sequence prediction component takes as input both the preconditions induced by the text component as well as the planning state and the previous subgoal. Thus ϕ_x contains features which check whether two subgoals are connected via an induced precondition relation, in addition to features which are simply the Cartesian product of domain predicates.

Words
Dependency Types
Dependency Type $\times$ Direction
Word $\times$ Dependency Type
Word $\times$ Dependency Type $\times$ Direction

Table 2: Example text features. A subgoal pair $\langle x_i, x_j \rangle$ is first mapped to word tokens using a small grounding table. Words and dependencies are extracted along paths between mapped target words. These are combined with path directions to generate the text features.

Domain Predicate	**Noun Phrases**
`have(plank)`	wooden plank, wood plank
`have(stone)`	stone, cobblestone
`have(iron)`	iron ingot

Table 3: Examples in our seed grounding table. Each predicate is mapped to one or more noun phrases that describe it in the text.

6 Experimental Setup

Datasets As the text description of our virtual world, we use documents from the Minecraft Wiki,[7] the most popular information source about the game. Our manually constructed seed grounding of predicates contains 74 entries, examples of which can be seen in Table 3. We use this seed grounding to identify a set of 242 sentences that reference predicates in the Minecraft domain. This results in a set of 694 *Candidate Relations*. We also manually annotated the relations expressed in the text, identifying 94 of the *Candidate Relations* as valid. Our corpus contains 979 unique word types and is composed of sentences with an average length of 20 words.

We test our system on a set of 98 problems that involve collecting resources and constructing objects in the Minecraft domain – for example, fishing, cooking and making furniture. To assess the complexity of these tasks, we manually constructed high-level plans for these goals and solved them using the Metric-FF planner. On average, the execution of the sequence of low-level plans takes 35 actions, with 3 actions for the shortest plan and 123 actions for the longest. The average branching factor is 9.7, leading to an average search space of more than 10^{34} possible action sequences. For evaluation purposes we manually identify a set of *Gold Relations* consisting of all precondition relations that are valid in this domain, including those not discussed in the text.

Evaluation Metrics We use our manual annotations to evaluate the type-level accuracy of relation extraction. To evaluate our high-level planner, we use the standard measure adopted by the IPC. This evaluation measure simply assesses whether the planner completes a task within a predefined time.

Baselines To evaluate the performance of our relation extraction, we compare against an SVM classifier[8] trained on the *Gold Relations*. We test the SVM baseline in a leave-one-out fashion.

To evaluate the performance of our text-aware high-level planner, we compare against five baselines. The first two baselines – *FF* and *No Text* – do not use any textual information. The *FF* baseline directly runs the Metric-FF planner on the given task, while the *No Text* baseline is a variant of our model that learns to plan in the reinforcement learning framework. It uses the same state-level features

[7]http://www.minecraftwiki.net/wiki/Minecraft_Wiki/

[8]SVM$^{\text{light}}$ (Joachims, 1999) with default parameters.

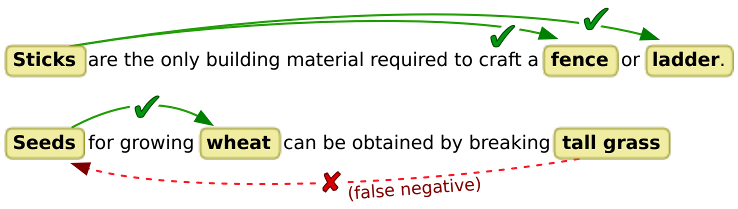
Sticks are the only building material required to craft a fence or ladder.
Seeds for growing wheat can be obtained by breaking tall grass
✓
✗
(false negative)

Figure 4: Examples of precondition relations predicted by our model from text. Check marks (✓) indicate correct predictions, while a cross (✗) marks the incorrect one – in this case, a valid relation that was predicted as invalid by our model. Note that each pair of highlighted noun phrases in a sentence is a *Candidate Relation*, and pairs that are not connected by an arrow were correctly predicted to be invalid by our model.

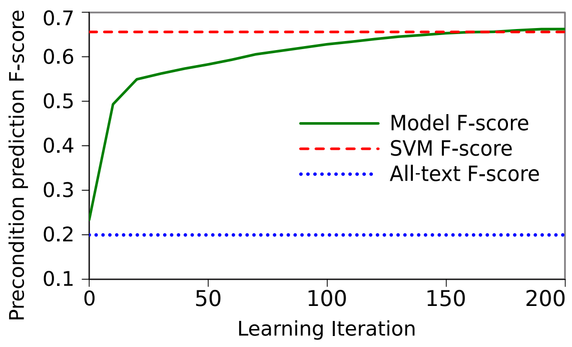

Figure 5: The performance of our model and a supervised SVM baseline on the precondition prediction task. Also shown is the F-Score of the full set of *Candidate Relations* which is used unmodified by *All Text*, and is given as input to our model. Our model's F-score, averaged over 200 trials, is shown with respect to learning iterations.

Method	**% Plans**
FF	40.8
No text	69.4
All text	75.5
Full model	**80.2**
Manual text	84.7
Gold connection	87.1

Table 4: Percentage of tasks solved successfully by our model and the baselines. All performance differences between methods are statistically significant at $p \leq .01$.

as our model, but does not have access to text.

The *All Text* baseline has access to the full set of 694 *Candidate Relations*. During learning, our full model refines this set of relations, while in contrast the *All Text* baseline always uses the full set.

The two remaining baselines constitute the upper bound on the performance of our model. The first, *Manual Text*, is a variant of our model which directly uses the links derived from manual annotations of preconditions in text. The second, *Gold*, has access to the *Gold Relations*. Note that the connections available to *Manual Text* are a subset of the *Gold* links, because the text does not specify all relations.

Experimental Details All experimental results are averaged over 200 independent runs for both our model as well as the baselines. Each of these trials is run for 200 learning iterations with a maximum subgoal sequence length of 10. To find a low-level plan between each consecutive pair of subgoals, our high-level planner internally uses Metric-FF. We give Metric-FF a one-minute timeout to find such a low-level plan. To ensure that the comparison between the high-level planners and the FF baseline is fair, the FF baseline is allowed a runtime of 2,000 minutes. This is an upper bound on the time that our high-level planner can take over the 200 learning iterations, with subgoal sequences of length at most 10 and a one minute timeout. Lastly, during learning we initialize all parameters to zero, use a fixed learning rate of 0.0001, and encourage our model to explore the state space by using the standard ϵ-greedy exploration strategy (Sutton and Barto, 1998).

7 Results

Relation Extraction Figure 5 shows the performance of our method on identifying preconditions in text. We also show the performance of the supervised SVM baseline. As can be seen, after 200 learning iterations, our model achieves an F-Measure of 66%, equal to the supervised baseline. These results support our hypothesis that planning feedback is a powerful source of supervision for analyzing a given text corpus. Figure 4 shows some examples of sentences and the corresponding extracted relations.

Planning Performance As shown in Table 4 our text-enriched planning model outperforms the text-free baselines by more than 10%. Moreover, the performance improvement of our model over the *All Text* baseline demonstrates that the accuracy of the

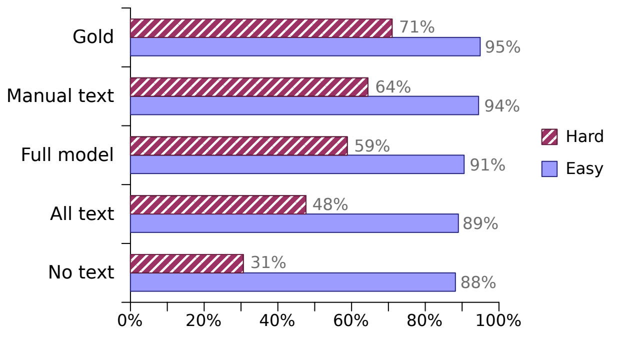

Figure 6: Percentage of problems solved by various models on Easy and Hard problem sets.

path has word "**use**"
path has word "**fill**"
path has dependency type "**dobj**"
path has dependency type "**xsubj**"
path has word "**craft**"

path has word "**craft**"
path has dependency type "**partmod**"
path has word "**equals**"
path has word "**use**"
path has dependency type "**xsubj**"

Figure 7: The top five positive features on words and dependency types learned by our model (above) and by SVM (below) for precondition prediction.

extracted text relations does indeed impact planning performance. A similar conclusion can be reached by comparing the performance of our model and the *Manual Text* baseline.

The difference in performance of 2.35% between *Manual Text* and *Gold* shows the importance of the precondition information that is missing from the text. Note that *Gold* itself does not complete all tasks – this is largely because the Markov assumption made by our model does not hold for all tasks.[9]

Figure 6 breaks down the results based on the difficulty of the corresponding planning task. We measure problem complexity in terms of the low-level steps needed to implement a manually constructed high-level plan. Based on this measure, we divide the problems into two sets. As can be seen, all of the high-level planners solve almost all of the easy problems. However, performance varies greatly on the more challenging tasks, directly correlating with planner sophistication. On these tasks our model outperforms the *No Text* baseline by 28% and the *All Text* baseline by 11%.

Feature Analysis Figure 7 shows the top five positive features for our model and the SVM baseline. Both models picked up on the words that indicate precondition relations in this domain. For instance, the word *use* often occurs in sentences that describe the resources required to make an object, such as “bricks are items used to craft brick blocks”. In addition to lexical features, dependency information is also given high weight by both learners. An example of this is a feature that checks for the direct object dependency type. This analysis is consistent with prior work on event semantics which shows lexicosyntactic features are effective cues for learning text relations (Blanco et al., 2008; Beamer and Girju, 2009; Do et al., 2011).

[9]When a given task has two non-trivial preconditions, our model will choose to satisfy one of the two first, and the Markov assumption blinds it to the remaining precondition, preventing it from determining that it must still satisfy the other.

8 Conclusions

In this paper, we presented a novel technique for inducing precondition relations from text by grounding them in the semantics of planning operations. While using planning feedback as its only source of supervision, our method for relation extraction achieves a performance on par with that of a supervised baseline. Furthermore, relation grounding provides a new view on classical planning problems which enables us to create high-level plans based on language abstractions. We show that building high-level plans in this manner significantly outperforms traditional techniques in terms of task completion.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, grant IIS-0835652), the DARPA Machine Reading Program (FA8750-09-C-0172, PO#4910018860), and Batelle (PO#300662). Thanks to Amir Globerson, Tommi Jaakkola, Leslie Kaelbling, George Konidaris, Dylan Hadfield-Menell, Stefanie Tellex, the MIT NLP group, and the ACL reviewers for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Fahiem Bacchus and Qiang Yang. 1994. Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intell.*, 71(1):43–100.

Jennifer L. Barry, Leslie Pack Kaelbling, and Toms Lozano-Prez. 2011. DetH*: Approximate hierarchical solution of large markov decision processes. In *IJCAI'11*, pages 1928–1935.

Brandon Beamer and Roxana Girju. 2009. Using a bigram event model to predict causal potential. In *Proceedings of CICLing*, pages 430–441.

Eduardo Blanco, Nuria Castell, and Dan Moldovan. 2008. Causal relation extraction. In *Proceedings of the LREC'08*.

S.R.K Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of ACL*, pages 82–90.

S.R.K Branavan, Luke Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of ACL*, pages 1268–1277.

S. R. K. Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a montecarlo framework. In *Proceedings of ACL*, pages 268–277.

Du-Seong Chang and Key-Sun Choi. 2006. Incremental cue phrase learning and bootstrapping method for causality extraction using cue phrase and word pair probabilities. *Inf. Process. Manage.*, 42(3):662–678.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

Q. Do, Y. Chan, and D. Roth. 2011. Minimally supervised event causality identification. In *EMNLP*, 7.

Michael Fleischman and Deb Roy. 2005. Intentional context in situated natural language learning. In *Proceedings of CoNLL*, pages 104–111.

Maria Fox and Derek Long. 2003. Pddl2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:2003.

Malik Ghallab, Dana S. Nau, and Paolo Traverso. 2004. *Automated Planning: theory and practice*. Morgan Kaufmann.

Roxana Girju and Dan I. Moldovan. 2002. Text mining for causal relations. In *Proceedigns of FLAIRS*, pages 360–364.

Jörg Hoffmann and Bernhard Nebel. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302.

Thorsten Joachims. 1999. Advances in kernel methods. chapter Making large-scale support vector machine learning practical, pages 169–184. MIT Press.

Anders Jonsson and Andrew Barto. 2005. A causal approach to hierarchical decomposition of factored mdps. In *Advances in Neural Information Processing Systems, 13:10541060*, page 22. Press.

Marián Lekavý and Pavol Návrat. 2007. Expressivity of strips-like and htn-like planning. *Lecture Notes in Artificial Intelligence*, 4496:121–130.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of ACL*, pages 91–99.

Neville Mehta, Soumya Ray, Prasad Tadepalli, and Thomas Dietterich. 2008. Automatic discovery and transfer of maxq hierarchies. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 648–655.

Raymond J. Mooney. 2008a. Learning language from its perceptual context. In *Proceedings of ECML/PKDD*.

Raymond J. Mooney. 2008b. Learning to connect language and perception. In *Proceedings of AAAI*, pages 1598–1601.

A. Newell, J.C. Shaw, and H.A. Simon. 1959. *The processes of creative thinking*. Paper P-1320. Rand Corporation.

James Timothy Oates. 2001. *Grounding knowledge in sensors: Unsupervised learning for language and planning*. Ph.D. thesis, University of Massachusetts Amherst.

Avirup Sil and Alexander Yates. 2011. Extracting STRIPS representations of actions and events. In *Recent Advances in Natural Language Learning (RANLP)*.

Avirup Sil, Fei Huang, and Alexander Yates. 2010. Extracting action and event semantics from web text. In *AAAI 2010 Fall Symposium on Commonsense Knowledge (CSK)*.

Jeffrey Mark Siskind. 2001. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, 15:31–90.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in NIPS*, pages 1057–1063.

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the ACL*, pages 806–814.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8.

Alicia P. Wolfe and Andrew G. Barto. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *In Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 816–823.

Chen Yu and Dana H. Ballard. 2004. On the integration of grounding language and learning objects. In *Proceedings of AAAI*, pages 488–493.