Reading to Learn: Constructing Features from Semantic Abstracts

Jacob Eisenstein[*] **James Clarke**[†] **Dan Goldwasser**[†] **Dan Roth**[*†]
[*]Beckman Institute for Advanced Science and Technology, [†]Department of Computer Science
University of Illinois
Urbana, IL 61801
{jacobe,clarkeje,goldwas1,danr}@illinois.edu

Abstract

Machine learning offers a range of tools for training systems from data, but these methods are only as good as the underlying representation. This paper proposes to acquire representations for machine learning by reading text written to accommodate human learning. We propose a novel form of semantic analysis called *reading to learn*, where the goal is to obtain a high-level semantic abstract of multiple documents in a representation that facilitates learning. We obtain this abstract through a generative model that requires no labeled data, instead leveraging repetition across multiple documents. The semantic abstract is converted into a transformed feature space for learning, resulting in improved generalization on a relational learning task.

1 Introduction

Machine learning offers a range of powerful tools for training systems to act in complex environments, but these methods depend on a well-chosen representation for features. For learning to succeed the representation often must be crafted with knowledge about the application domain. This poses a bottleneck, requiring expertise in both machine learning and the application domain. However, domain experts often express their knowledge through text; one direct expression is through text designed to aid human learning. In this paper we exploit text written by domain experts in order to build a more expressive representation for learning. We term this approach *reading to learn*.

The following scenario demonstrates the motivation for reading to learn. Imagine an agent given a task within its world/environment. The agent has no prior knowledge of the task but can perceive the world through low-level sensors. Learning directly from the sensors may be difficult, as interesting tasks typically require a complex combination of sensors. Our goal is to acquire domain knowledge through the semantic analysis of text, so as to produce higher-level relations through combinations of sensors.

As a concrete example consider the problem of learning how to make legal moves in Freecell solitaire. Relevant sensors may indicate if an object is a card or a freecell, whether a card is a certain value, and whether two values are in sequence. Although it is possible to express the rules with a combination of sensors, learning this combination is difficult. Text can facilitate learning by providing relations at the appropriate level of generalization. For example, the sentence: "You can place a card on an empty freecell," suggests not only which sensors are useful together but also how these sensors should be linked. Assuming the sensors are represented as predicates, one possible relation this sentence suggests is: $\texttt{r}(x, y) = \texttt{card}(x) \wedge \texttt{freecell}(y) \wedge \texttt{empty}(y)$. Armed with this new relation the agent's learning task may be simpler. Throughout the paper we refer to low-level sensory input as *sensor* or *predicate*, and to a higher level concept as a *logical formula* or *relation*.

Our approach to semantic analysis does not require a complete semantic representation of the text. We merely wish to acquire a *semantic abstract* of a document or document collection, and use the discovered relations to facilitate data-driven learning. This will allow us to directly evaluate the contribution of the extracted relations for learning.

We develop an approach to recover semantic abstracts that uses minimal supervision: we assume only a very small set of lexical *glosses*, which map from words to sensors. This marks a substantial departure from previous work on semantic parsing, which requires either annotations of the meanings of each individual sentence (Zettlemoyer and Collins, 2005; Liang et al., 2009), or alignments of sentences to grounded representations of the

world (Chen and Mooney, 2008). For the purpose of learning, this approach may be inapplicable, as such text is often written at a high level of abstraction that permits no grounded representation.

There are two properties of our setting that make unsupervised learning feasible. First, it is not necessary to extract a semantic representation of each individual sentence, but rather a summary of the semantics of the document collection. Errors in the semantic abstract are not fatal, as long it guides the learning component towards a more useful representation. Second, we can exploit repetition across documents, which should generally express the same underlying meaning. Logical formulae that are well-supported by multiple documents are especially likely to be useful.

The rest of this paper describes our approach for recovering semantic abstracts and outlines how we apply and evaluate this approach on the Freecell domain. The paper contributes the following key ideas: (1) Interpreting text which is written at a level of abstraction where no grounded representation is possible, (2) *reading to learn*, a new setting in which extracted semantic representations are evaluated by whether they facilitate learning; (3) *abstractive semantic summarization*, aimed at capturing broad semantic properties of a multi-document dataset, rather than a semantic parse of individual sentences; (4) a novel, minimally-supervised generative model for semantic analysis which leverages both lexical and syntactic properties of text.

2 Approach Overview

We describe our approach to text analysis as *multidocument semantic abstraction*, with the goal of discovering a compact set of logical formulae to explain the text in a document collection. To this end, we develop a novel generative model in which natural language sentences (e.g., "You can always place cards in empty freecells") are stochastically generated from logical formulae (e.g., $\texttt{card}(x) \wedge \texttt{freecell}(y) \wedge \texttt{empty}(y)$). We formally define a generative process that reflects our intuitions about the relationship between formulae and sentences (Section 3), and perform sampling-based inference to recover the formulae most likely to have generated the observed data (Section 4). The top N such formulae can then be added as additional predicates for relational learning.

Our semantic representation consists of conjunctions of *literals*, each of which includes a single *predicate* (e.g., `empty`) and one or more *variables* (e.g., x). Predicates describe atomic semantic concepts, while variables construct networks of relationships between them. While the importance of the predicates is obvious, the variable assignments also exert a crucial influence on the semantics of the conjunction: modifying a single variable in the formula above from $\texttt{empty}(y)$ to $\texttt{empty}(x)$ yields a formula that is trivially false for all groundings (since cards can never be empty).

Thus, our generative model must account for the influence of both predicates and variables on the sentences in the documents. A natural choice is to use the predicates to influence the lexical items, while letting the variables determine the syntactic structure. For example, the formula $\texttt{card}(x) \wedge \texttt{freecell}(y) \wedge \texttt{empty}(y)$ contains three predicates and two variables. The predicates influence the lexical items in a direct way: we expect that sentences generated from this formula will include a member of the gloss set for each predicate – the sentence "Put the cards on the empty freecells" should be more likely than "Columns are constructed by playing cards in alternating colors."

The impact of the variables on the generative process is more subtle. The sharing of the variable y suggests a relationship between the predicates `freecell` and `empty`. This should be realized in the syntactic structure of the sentence. Modeling syntax using a dependency tree, we expect that the glosses for predicates that share terms will appear in compact sub-trees, while predicates that do not share terms should be more distant. One possible surface realization of this logical formula is the sentence, "Put the card on the empty freecell," whose dependency parse is shown in the left tree of Figure 1. The glosses *empty* and *freecell* are immediately adjacent, while *card* is more remote.

We develop two metrics that quantify the compactness of a set of variable assignments with respect to a dependency tree: *excess terms*, and *shared terms*. The number of excess terms in a subtree is the number of unique terms assigned to words in the subtree, minus the maximum arity of any predicate in the subtree. Shared terms arise whenever a node has multiple subtrees which each contain the same variable. We will use the alternative alignments in Figure 1 to provide a more detailed explanation. In each tree, the variables are written in the nodes belonging to the associated lexical items; variables are written over arrows to indicate membership in some node in the subtree.

Excess Terms Alignment A of Figure 1, corresponding to the formula $\texttt{card}(x) \wedge \texttt{freecell}(x) \wedge \texttt{empty}(x)$, has zero

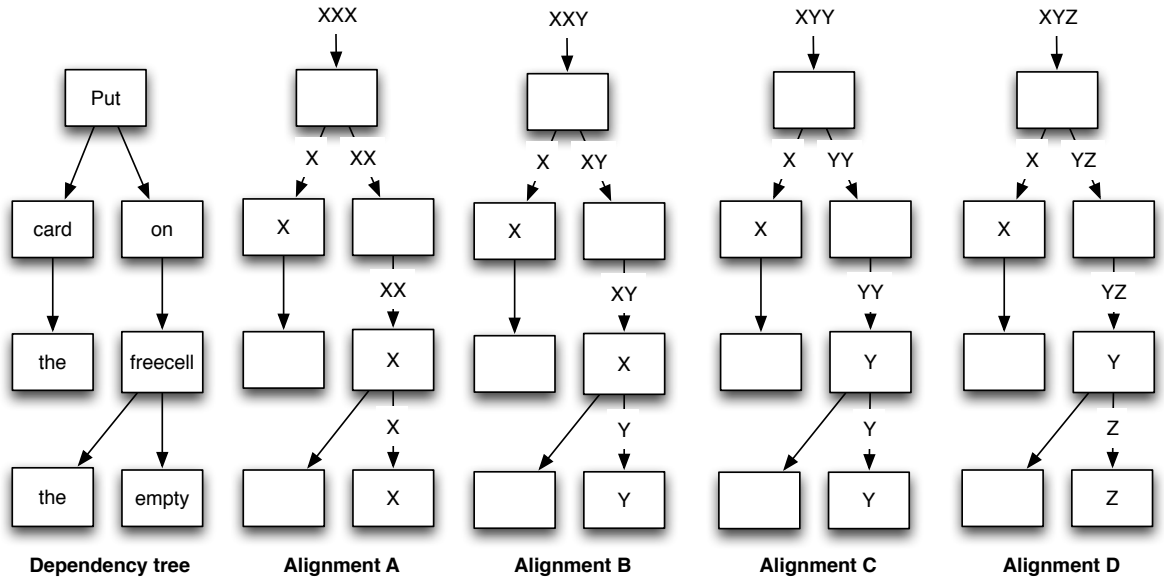

Figure 1: A dependency parse and four different variable assignments. Each literal is aligned to a word (a node in the graph), and the associated variables are written in the box. Variables belonging to descendant nodes are written over the arrows.

excess terms in every subtree; there is a total of one variable, and all the predicates are unary. In Alignment B, $\texttt{card}(x) \wedge \texttt{freecell}(x) \wedge \texttt{empty}(y)$, there are excess terms at the root, and in the top two subtrees on the right-hand side. Alignment C contains an excess term at only the root node. Even though it contains the same number of unique variables as Alignment B, it is not penalized as harshly because the alignment of variables better corresponds to the syntactic structure. Alignment D contains the greatest number of excess terms: two at the root of the tree, and one in each of the top two subtrees on the right side.

Shared Terms According to the excess term metric, the best choice is simply to introduce as few variables as possible. For this reason, we also penalize *shared terms* which occur when a node has subtree children that share a variable. In Figure 1, Alignments A and B each contain a shared term at the top node; Alignments C and D contain no shared terms.

Overall, we note that Alignment B is penalized on both metrics, as it contains both excess terms and shared terms; the syntactic structure of the sentence makes such a variable assignment relatively improbable.

3 Generative Model

These intuitions are formalized in a generative account of how sentences are stochastically produced from a set of logical formulae. This generative story guides an inference procedure for recovering logical formulae that are likely to have generated any observed set of texts, which is described in Section 4.

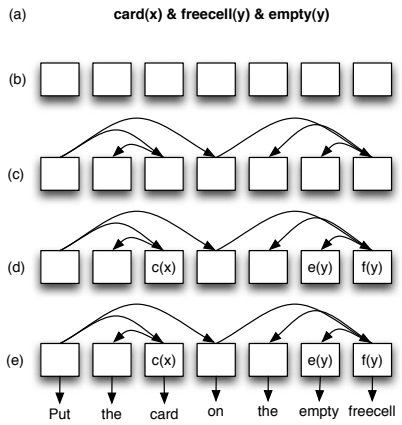

Figure 2: A graphical depiction of the generative process by which sentences are produced from formulae

The outline of the generative process is depicted in Figure 2. For each sentence, we begin in step (a) by drawing a formula f from a Dirichlet process (Ferguson, 1973). The Dirichlet process defines a non-parametric mixture model, and has the effect of adaptively selecting the appropriate number of formulae to explain the observed sentences

in the corpus.[1] We then draw the sentence length from some distribution over positive integers; as the sentence length is always observed, we need not define the distribution (step (b)). In step (c), a dependency tree is drawn from a uniform distribution over spanning trees with a number of nodes equal to the length of the sentence. In step (d) we draw an alignment of the literals in f to nodes in the dependency tree, written $\mathbf{a}_t(f)$. The distribution over alignments is described in Section 3.1. Finally, the aligned literals are used to generate the words at each slot in the dependency tree. A more formal definition of this process is as follows:

- Draw λ, the expected number of literals per formula, from a Gamma distribution $\mathcal{G}(u, v)$.
- Draw an infinite set of formulae $\mathbf{f}$. For each formula f_i,
  - Draw the formula length $\#|f_i|$ from a Poisson distribution, $n_i \sim \text{Poisson}(\lambda)$.
  - Draw n_i literals from a uniform distribution.
- Draw π, an infinite multinomial distribution over formulae: $\pi \sim \text{GEM}(\pi_0)$, where GEM refers to the stick-breaking prior (Sethuraman, 1994) and $\pi_0 = 1$ is the concentration parameter. By attaching the multinomial π to the infinite set of formulae $\mathbf{f}$, we create a Dirichlet process. This is conventionally written $DP(\pi_0, G_0)$, where the base distribution G_0 encodes only the distribution over the number of literals, Poisson(λ).
- For each of D documents, draw the number of sentences $T \sim$ Poisson. For each of the T sentences in the document,
  - Draw a formula $f \sim DP(\pi_0, G_0)$ from the Dirichlet Process described above.
  - Draw a sentence length $\#|s| \sim$ Poisson.
  - Draw a dependency graph t (a spanning tree of size $\#|s|$) from a uniform distribution.
  - Draw an *alignment* $\mathbf{a}_t(f)$, an injective mapping from literals in f to nodes in the dependency structure t. The distribution over alignments is described in Section 3.1.
  - Draw the sentence s from the formula f and the alignment $\mathbf{a}(f)$. For each word token $w_i \in s$ is drawn from $p(w_i|a_t(f,i))$, where $a_t(f,i)$ indicates the (possibly empty) literal assigned to slot i in the alignment $\mathbf{a}_t(f)$ (Section 3.2).

[1] There are many recent applications of Dirichlet processes in natural language processing, e.g. Goldwater et al. (2006).

3.1 Distribution over Alignments

The distribution over alignments reflects our intuition that when literals share variables, they will be aligned to word slots that are nearby in the dependency structure; literals that do not share variables should be more distant. This is formalized by applying the concepts of excess terms and shared terms defined in Section 2. After computing the number of excess and shared terms in each subtree t_i, we can compute a local score (LS) for that subtree:

$$LS(\mathbf{a}_t(f); t_i) = \alpha \cdot NShared(\mathbf{a}_t(f), t_i) + \beta \cdot NExcess(\mathbf{a}_t(f), t_i) \cdot height(t_i).$$

This scoring function can be applied recursively to each subtree in t; the overall score of the tree is the recursive sum,

$$score(\mathbf{a}_t(f); t) = LS(\mathbf{a}_t(f); t) + \sum_i^n score(\mathbf{a}_t(f); t_i), \quad (1)$$

where t_i indicates the i^{th} subtree of t. We hypothesize a generative process that produces all possible alignments, scores them using $score(\mathbf{a}_t(f); t)$, and selects an alignment with probability,

$$p(\mathbf{a}_t(f)) \propto \exp\{-score(\mathbf{a}_t(f); t)\}. \quad (2)$$

In our experiments, we define the parameters $\alpha = 1, \beta = 1$.

3.2 Generation of Lexical Items

Once the logical formula is aligned to the parse structure, the generation of the lexical items in the sentence is straightforward. For word slots to which no literals are aligned, the lexical item is drawn from a language model θ, estimated from the entire document collection. For slots to which at least one literal is aligned, we construct a language model ϕ in which the probability mass is divided equally among all glosses of aligned predicates. The language model θ is used as a backoff, so that there is a strong bias in favor of generating glosses, but some probability mass is reserved for the other lexical items.

4 Inference

This section describes a sampling-based inference procedure for obtaining a set of formulae $\mathbf{f}$ that explain the observed text $\mathbf{s}$ and dependency structures $\mathbf{t}$. We perform Gibbs sampling over the formulae assigned to each sentence. Using the Chinese Restaurant Process interpretation of the Dirichlet Process (Aldous, 1985), we marginalize π, the infinite multinomial over all possible formulae: at each sampling step we select either an existing formula, or stochastically generate a new formula. After each full round of Gibbs sampling, a set of Metropolis-Hastings moves are applied to explore modifications of the formulae. This procedure converges on a stationary Markov chain centered on a set of formulae that cohere well with the lexical and syntactic properties of the text.

4.1 Assigning Sentences to Formulae

For each sentence s_i and dependency tree t_i, a hidden variable y_i indicates the index of the formula that generates the text. We can resample y_i using Gibbs sampling. In the non-parametric setting, y_i ranges over all non-negative integers; the Chinese Restaurant Process formulation marginalizes the infinite-dimensional parameter π, yielding a prior based on the counts for each "active" formula (to which at least one other sentence is assigned), and a pseudo-count representing all non-active formulae. Given K formulae, the prior on selecting formula j is:

$$p(y_i = j|\mathbf{y}_{-i}, \pi_0) \propto \begin{cases} \mathbf{n}_{-i}(j) & j < K \\ \pi_0 & j = K, \end{cases} \quad (3)$$

where $\mathbf{y}_{-i}$ refers to the assignments of all y other than y_i and $\mathbf{n}_{-i}$ refers to the counts over these assignments. Each $j < K$ identifies an existing formula in $\mathbf{f}$, to which at least one other sentence is assigned. When $j = K$, this means a new formula f^* must be generated.

To perform Gibbs sampling, we draw from the posterior distribution over y_i,

$$p(y_i|s_i, t_i\mathbf{f}, f^*, \mathbf{y}_{-i}, \pi_0) \propto p(y_i|\mathbf{y}_{-i}, \pi_0)p(s_i, t_i|\mathbf{y}_i, \mathbf{f}, f^*),$$

where the first term is the prior defined in Equation 3 and the latter term is the likelihood of generating the parsed sentence $\langle s_i, t_i \rangle$ from the formula indexed by y_i.

To compute the probability of a parsed sentence given a formula, we sum over alignments,

$$\begin{aligned} p(s, t|f) &= \sum_{\mathbf{a}_t(f)} p(s, t, \mathbf{a}_t(f)|f) \\ &= \sum_{\mathbf{a}_t(f)} p(s|\mathbf{a}_t(f))p(t, \mathbf{a}_t(f)|f) \\ &= \sum_{\mathbf{a}_t(f)} p(s|\mathbf{a}_t(f))p(\mathbf{a}_t(f)|t, f)p(t|f), \end{aligned} \quad (4)$$

applying the chain rule and independence assumptions from the generative model. The result is a product of three terms: the likelihood of the lexical items given the aligned predicates (defined in Section 3.2; the likelihood of the alignment given the dependency tree and formula (defined in equation 2), and the probability of the dependency tree given the formula, which is uniform.

Equation 4 takes a sum across alignments, but most of the probability mass of $p(s|\mathbf{a}_t(f))$ will be concentrated on alignments in which predicates cover words that gloss them. Thus, we can apply an approximation,

$$p(s, t|f) \approx \sum_{\mathbf{a}_t(f)}^{N} p(s|\mathbf{a}_t(f))p(\mathbf{a}_t(f)|t, f)p(t|f), \quad (5)$$

in which we draw N samples in which predicates are aligned to their glosses whenever possible.

Similarly, Equation 2 quantifies the likelihood of an alignment only to a constant of proportionality; again, a sum over possible alignments is necessary. We do not expect the prior on alignments to be strongly peaked like the sentence likelihood, so we approximate the normalization term by sampling M alignments at random and extrapolating:

$$\begin{aligned} p(\mathbf{a}_t(f)|t, f) &\propto q(\mathbf{a}_t(f); t) \\ &= \frac{q(\mathbf{a}_t(f); t)}{\sum_{\mathbf{a}'_t(f)} q(\mathbf{a}'_t(f); t)} \\ &\approx \frac{\#|\mathbf{a}'_t(f)|}{M} \frac{q(\mathbf{a}_t(f); t)}{\sum_{\mathbf{a}'_t(f)}^{M} q(\mathbf{a}'_t(f); t)}, \end{aligned}$$

where $q(\mathbf{a}_t(f); t) = \exp\{-score(\mathbf{a}_t(f); t)\}$, defined in Equation 2. In our experiments, we set N to at most 10, and $M = 20$. Drawing larger numbers of samples had no discernible effect on system output.

4.1.1 Generating new formulae

Chinese Restaurant Process sampling requires the generation of new candidate formulae at each resampling stage. To generate a new formula, we first sample the number of literals. As described in the generative story (Section 3), the number of literals is drawn from a Poisson distribution with parameter θ. We treat θ as unknown and marginalize, using the Gamma hyperprior $\mathcal{G}(u, v)$. Due to Poisson-Gamma conjugacy, this marginalization can be performed analytically, yielding a Negative-Binomial distribution with parameters $\langle u + \sum_i \#|f_i|, (1 + K + v)^{-1} \rangle$, where $\sum_i \#|f_i|$ is the sum of the number of literals in each formula, and K is the number of formulae which generate at least one sentence. In this sense, the hyperpriors u and v act as pseudo counts. We set $u = 3, v = 1$, reflecting a weak prior expectation of three literals per predicate.

After drawing the size of the formula, the predicates are selected from a uniform random distribution. Finally, the terms are assigned: at each slot, we reuse a previous term with probability 0.5, unless none is available; otherwise a new term is generated.

4.2 Proposing changes to formulae

The assignment resampling procedure has the ability to generate new formulae, thus exploring the space of relational features. However, to explore this space more rapidly, we introduce four Metropolis-Hastings moves that modify existing formulae (Gilks, 1995): adding a literal, deleting a literal, substituting a literal, and rearranging the terms of the formula. For each proposed move, we recompute the joint likelihood of the formula and all aligned sentences. The move is stochastically accepted based on the ratio of the joint likelihoods of the new and old configurations, multiplied by a Hastings correction.

The joint likelihood with respect to formula f is computed as $p(\mathbf{s}, \mathbf{t}, f) = p(f) \prod_i p(s_i, t_i|f)$. The prior on f considers only the number of literals, using a Negative-Binomial distribution as described in section 4.1.1. The likelihood $p(s_i, t_i|f)$ is given in equation 4. The Hastings correction is $\tilde{p}(f' \rightarrow f)/\tilde{p}(f \rightarrow f')$, with $\tilde{p}(f \rightarrow f')$ indicating the probability of proposing a move from f to f',and $\tilde{p}(f' \rightarrow f)$ indicating the probability of proposing the reverse move. The Hastings corrections depend on the arity of the predicates being added and removed; the derivation is straightforward but tedious. We plan to release a technical report with complete details.

4.3 Summary of inference

The final inference procedure iterates between Gibbs sampling of assignments of formulae to sentences, and manipulating the formulae through Metropolis-Hastings moves. A full iteration comprises proposing a move to each formula, and then using Gibbs sampling to reconsider all assignments. If a formula no longer has any sentences assigned to it, then it is dropped from the active set, and can no longer be selected in Gibbs sampling – this is standard in the Chinese Restaurant Process.

Five separate Markov chains are maintained in parallel. To allow the sampling procedure to converge to a stationary distribution, each chain begins with 100 iterations of "burn-in" sampling, without storing the output. At this point, we perform another 100 iterations, storing the state at the end of each iteration.[2] All formulae are ranked according to the cumulative number of sentences to which they are assigned (across all five Markov chains), aggregating the counts for multiple instances of identical formulae. This yields a ranked list of formulae which will be used in our framework as features for relational learning.

5 Evaluation

Our experimental setup is designed to evaluate the quality of the semantic abstraction performed by our model. The logical formulae obtained by our system are applied as features for relational learning of the rules of the game of Freecell solitaire. We investigate whether these features enable better generalization given varying number of training examples of Freecell game states. We also quantify the specific role of syntax, lexical choice, and feature expressivity in learning performance. This section describes the details of this evaluation.

5.1 Relational Learning

We perform relational learning using Inductive Logic Programming (ILP), which constructs generalized rules by assembling smaller logical formulae to explain observed propositional examples (Muggleton, 1995). The lowest level formulae consist of basic sensors that describe the environment. ILP's expressivity enables it to build complex conjunctions of these building blocks, but at the cost of tractability. Our evaluation asks whether the logical formulae abstracted from text

[2] Sampling for more iterations was not found to affect performance on development data, and the model likelihood appeared stationary after 100 iterations.

Predicate	Glosses
card(x)	card
tableau(x)	column, tableau
freecell(x)	freecell, cell
homecell(x)	foundation, cell, homecell
value(x,y)	ace, king, rank, 8, 3, 7, lowest, highest
successor(x,y)	higher, sequence, sequential
color(x,y)	black, red, color
suit(x,y)	suit, club, diamond, spade, heart
on(x,y)	onto
top(x,y)	bottom, available, top
empty(x)	empty

Table 1: Predicates in the Freecell world model, with natural language glosses obtained from the development set text.

can transform the representation to facilitate learning. We compare against both the sensor-level representation as well as richer representations that do not benefit from the full power of our model's semantic analysis.

The ALEPH[3] ILP system, which is primarily based on PROGOL (Muggleton, 1995), was used to induce the rules of game. The search parameters remained constant for all experiments.

5.2 Resources

There are four types of resources required to work in the reading-to-learn setting: a world model, instructional text, a small set of glosses that map from text to elements of the world model, and labeled examples of correct and incorrect actions in the world. In our experiments, we consider the domain of Freecell solitaire, a popular card game (Morehead and Mott-Smith, 1983) in which cards are moved between various types of locations, depending on their suit and rank. We now describe the resources for the Freecell domain in more detail.

World Model Freecell solitaire can be described formally using first order logic; we consider a slightly modified version of the representation from the Planning Domain Definition Language (PDDL), which is used in automatic game-playing competitions. Specifically, there are 87 constants: 52 cards, 16 locations, 13 values, four suits, and two colors. These constants are combined with a fixed set of 11 predicates, listed in Table 1.

Instructional Text Our approach relies on text that describes how to operate in the Freecell solitaire domain. A total of five instruction sets were obtained from the Internet. Due to the popularity of the Microsoft implementation of Freecell, instructions often contain information specific to playing Freecell on a computer. We manually removed sentences which did not focus on the card aspects of Freecell (e.g., how to set up the board and information regarding where to click to move cards). In order to use our semantic abstraction model, the instructions were part-of-speech tagged with the Stanford POS Tagger (Toutanova and Manning, 2000) and dependency parses were obtained using Malt (Nivre, 2006).

Glosses Our reading to learn setting requires a small set of glosses, which are surface forms commonly used to represent predicates from the world model. We envision an application scenario in which a designer manually specifies a few glosses for each predicate. However, for the purposes of evaluation, it would be unprincipled for the experimenters to handcraft the ideal set of glosses. Instead, we gathered a development set of text and annotated the lexical mentions of the world model predicates in text. This annotation is used to obtain glosses to apply to the evaluation text. This approximates a scenario in which the designer has a reasonable idea of how the domain will be described in text, but no prior knowledge of the specific details of the text instructions. Our experiments used glosses that occurred two or more times in the instructions: this yields a total of 32 glosses for 11 predicates, as shown in Table 1.

Evaluation game data Ultimately, the semantic abstraction obtained from the text is applied to learning on labeled examples of correct and incorrect actions in the world model. For evaluation, we automatically generated a set of *move scenarios*: game states with one positive example (a legal move) and one negative example (an illegal move). To avoid bias in the data we generate an equal number of move scenarios from each of three types: moves to the freecells, homecells, and tableaux. For our experiments we vary the number of move scenarios in the training set; the development and test sets consist of 900 and 1500 move scenarios respectively.

5.3 Evaluation Settings

We compare four different feature sets, which will be provided to the ALEPH ILP learner. All feature sets include the sensor-level predicates shown in Table 1. The FULL-MODEL feature set also includes the top logical formulae obtained in our model's semantic abstract (see Sec-

[3]Freely available from http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/

tion 4.3). The NO-SYNTAX feature set is obtained from a variant of our model in which the influence of syntax is removed by setting parameters $\alpha, \beta = 0$. The SENSORS-ONLY feature set uses only the sensor-level predicates. Finally, the RELATIONAL-RANDOM feature set is constructed by replacing each feature in the FULL-MODEL set with a randomly generated relational feature of identical expressivity (each predicate is replaced by a randomly chosen alternative with identical arity; terms are also assigned randomly). This ensures that any performance gains obtained by our model were not due merely to the greater expressivity of its relational features. The number of features included in each scenario is tuned on a development set of test examples.

The performance metric assesses the ability of the ILP learner to classify proposed Freecell moves as legal or illegal. As the evaluation set contains an equal number of positive and negative examples, accuracy is the appropriate metric. The training scenarios are randomly generated; we repeat each run 50 times and average our results. For the RELATIONAL-RANDOM feature set – in which predicates and terms are chosen randomly – we also regenerate the formulae per run.

6 Results

Table 2 shows a comparison of the results using the setup described above. Our FULL-MODEL achieves the best performance at every training set size, consistently outperforming the SENSORS-ONLY representation by an absolute difference of three to four percent. This demonstrates the semantic abstract obtained by our model does indeed facilitate machine learning in this domain.

RELATIONAL-RANDOM provides a baseline of relational features with equal expressivity to those chosen by our model, but with the predicates and terms selected randomly. We consistently outperform this baseline, demonstrate that the improvement obtained over the sensors only representation is not due merely to the added expressivity of our features.

The third row compares against NO-SYNTAX, a crippled version of our model that incorporates lexical features but not the syntactic structure. The results are stronger than the SENSORS-ONLY and RELATIONAL-RANDOM baselines, but still weaker than our full system. This demonstrates the syntactic features incorporated by our model result in better semantic representations of the underlying text.

Features	Number of training scenarios			
	15	30	60	120
SENSORS-ONLY	79.12	88.07	92.77	93.73
RELATIONAL-RANDOM	82.72	89.14	93.08	94.17
NO-SYNTAX	80.98	89.79	94.11	97.04
FULL-MODEL	**82.89**	**91.00**	**95.23**	**97.45**

Table 2: Results as number of training examples varied. Each value represents the accuracy of the induced rules obtained with the given feature set.

$\texttt{card}(x_1) \wedge \texttt{tableau}(x_2)$
$\texttt{card}(x_1) \wedge \texttt{freecell}(x_2)$
$\texttt{homecell}(x_1) \wedge \texttt{value}(x_2,x_3)$
$\texttt{empty}(x_1) \wedge \texttt{freecell}(x_1)$
$\texttt{card}(x_1) \wedge \texttt{top}(x_1,x_2)$
$\texttt{card}(x_1) \wedge \texttt{homecell}(x_2)$
$\texttt{freecell}(x_1) \wedge \texttt{homecell}(x_2)$
$\texttt{card}(x_1) \wedge \texttt{tableau}(x_1)$
$\texttt{card}(x_1) \wedge \texttt{top}(x_2,x_1)$
$\texttt{homecell}(x_1)$
$\texttt{card}(x_1) \wedge \texttt{homecell}(x_1)$
$\texttt{color}(x_1,x_2) \wedge \texttt{value}(x_3,x_4)$
$\texttt{suit}(x_1,x_2) \wedge \texttt{value}(x_3,x_4)$
$\texttt{value}(x_1,x_2) \wedge \texttt{value}(x_3,x_4)$
$\texttt{homecell}(x_1) \wedge \texttt{successor}(x_2,x_3)$

Figure 3: The top 15 features recovered by the semantic abstraction of our full model.

Figure 3 shows the top 15 formulae recovered by the full model running on the evaluation text. Features such as $\texttt{empty}(x_1) \wedge \texttt{freecell}(x_1)$ are useful because they reuse variables to ensure that objects have key properties – in this case, ensuring that a freecell is empty. Other features, such as $\texttt{homecell}(x_1) \wedge \texttt{value}(x_2, x_3)$, help to focus the search on useful conjunctions of predicates (in Freecell, the legality of playing a card on a homecell depends on the value of the card). Note that three of these 15 formulae are trivially useless, in that they are always false: e.g., $\texttt{card}(x_1) \wedge \texttt{tableau}(x_1)$. This illustrates the importance of term assignment in obtaining useful features for learning. In the NO-SYNTAX system, which ignores the relationship between term assignment and syntactic structure, eight of the top 15 formulae were trivially useless due to term incompatibility.

7 Related Work

This paper draws on recent literature on extracting logical forms from surface text (Zettlemoyer and Collins, 2005; Ge and Mooney, 2005; Downey et al., 2005; Liang et al., 2009), interpreting language in the context of a domain (Chen and Mooney, 2008), and using an actionable domain to guide text interpretation (Branavan et al., 2009). We differentiate our research in several dimensions:

Language Interpretation Instructional text describes generalized statements about entities in the domain and the way they interact, thus the text does not correspond directly to concrete sensory inputs in the world (i.e., a specific world state). Our interpretation captures these generalizations as first-order logic statements that can be evaluated given a specific state. This contrasts to previous work which assumes a direct correspondence between text and world state (Branavan et al., 2009; Chen and Mooney, 2008).

Supervision Our work avoids supervision in the form of labeled examples, using only a minimal set of natural language glosses per predicate. Previous work also considered the supervision signal obtained by interpreting natural language in the context of a formal domain. Branavan et al. (2009) use feedback from a world model as a supervision signal. Chen and Mooney (2008) use temporal alignment of text and grounded descriptions of the world state. In these approaches, concrete domain entities are grounded in language interpretation, and therefore require only a propositional semantic representation. Previous approaches for interpreting generalized natural language statements are trained from labeled examples (Zettlemoyer and Collins, 2005; Lu et al., 2008).

Level of analysis We aim for an *abstractive semantic summary* across multiple documents, whereas other approaches attempt to produce logical forms for individual sentences (Zettlemoyer and Collins, 2005; Ge and Mooney, 2005). We avoid the requirement that each sentence have a meaningful interpretation within the domain, allowing us to handle relatively unstructured text.

Evaluation We do not evaluate the representations obtained by our model; rather we assess whether these representations improve learning performance. This is similar to work on GeoQuery (Wong and Mooney, 2007; Ge and Mooney, 2005), and also to recent work on following step-by-step directions (Branavan et al., 2009). While these evaluations are performed on the basis of individual sentences, actions, or system responses, we evaluate the holistic semantic analysis obtained by our system.

Model We treat surface text as generated from a latent semantic description. Lu et al. (2008) apply a generative model, but require a complete derivation from semantics to the lexical representation, while we favor a more flexible semantic analysis that can be learned without annotation and applied to noisy text. More similar is the work of Liang et al. (2009), which models the generation of semantically-relevant fields using lexical and discourse features. Our approach differs by accounting for syntax, which enables a more expressive semantic representation that includes ungrounded variables.

Relational learning The output of our semantic analysis is applied to learning in a structured relational space, using ILP. A key difficulty with ILP is that the increased expressivity dramatically expands the hypothesis space, and it is widely agreed that some learning bias is required for ILP to be tractable (Nédellec et al., 1996; Cumby and Roth, 2003). Our work can be viewed as a new method for acquiring such bias from text; moreover, our approach is not specialized for ILP and may be used to transform the feature space in other forms of relational learning as well (Roth and Yih, 2001; Cumby and Roth, 2003; Richardson and Domingos, 2006).

8 Conclusion

This paper demonstrates a new setting for semantic analysis, which we term *reading to learn*. We handle text which describes the world in general terms rather than refereing to concrete entities in the domain. We obtain a semantic abstract of a multiple document collection, using a novel, minimally-supervised generative model that accounts for both syntax and lexical choice. The semantic abstract is represented as a set of predicate logic formulae, which are applied as higher-order features for learning. We demonstrate that these features improve learning performance, and that both the lexical and syntactic aspects of our model yield substantial contributions.

In the current setup, we produce an "overgenerated" semantic representation comprised of useful features for learning but also some false positives. The learning aspect of our system can be seen as pruning this representation by selecting useful formulae based on interaction with the training data. In the future we hope to explore ways to interleave semantic analysis with exploration of the learning domain, thus "closing the loop" and using the environment as a supervision signal for linguistic analysis.

Acknowledgments We thank Gerald DeJong, Julia Hockenmaier, Alex Klementiev and the anonymous reviewers for their helpful feedback. This work is supported by DARPA funding under the Bootstrap Learning Program and the Beckman Institute Postdoctoral Fellowship.

References

Aldous, David J. 1985. Exchangeability and related topics. *Lecture Notes in Math* 1117:1–198.

Branavan, S. R. K., Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing Processing (ACL-IJCNLP 2009)*. Singapore.

Chen, David L. and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of 25th International Conference on Machine Learning (ICML 2008)*. Helsinki, Finland, pages 128–135.

Cumby, Chad and Dan Roth. 2003. On kernel methods for relational learning. In *Proceedings of the Twentieth International Conference (ICML 2003)*. Washington, DC, pages 107–114.

Downey, Doug, Oren Etzioni, and Stephen Soderland. 2005. A probabilistic model of redundancy in information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2005)*. pages 1034–1041.

Ferguson, Thomas S. 1973. A bayesian analysis of some nonparametric problems. *The Annals of Statistics* 1(2):209–230.

Ge, Ruifang and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. Ann Arbor, MI, pages 128–135.

Gilks, Walter R. 1995. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC.

Goldwater, Sharon, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*. Sydney, Australia, pages 673–680.

Liang, Percy, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing Processing (ACL-IJCNLP 2009)*. Singapore.

Lu, Wei, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2008)*. Honolulu, Hawaii, pages 783–792.

Morehead, Albert H. and Geoffrey Mott-Smith. 1983. *The Complete Book of Solitaire and Patience Games*. Bantam.

Muggleton, Stephen. 1995. Inverse entailment and progol. *New Generation Computing Journal* 13:245–286.

Nédellec, C., C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. 1996. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, IOS Press, pages 82–103.

Nivre, Joakim. 2006. *Inductive dependency parsing*. Springer.

Richardson, Matthew and Pedro Domingos. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Roth, Dan and Wen-tau Yih. 2001. Relational learning via propositional algorithms: An information extraction case study. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2001)*. pages 1257–1263.

Sethuraman, Jayaram. 1994. A constructive definition of dirichlet priors. *Statistica Sinica* 4(2):639–650.

Toutanova, Kristina and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*. pages 63–70.

Wong, Yuk Wah and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*. Prague, Czech Republic, pages 128–135.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*. pages 658–666.