A Study on Discriminative Learning of Weighted Abduction

B1IM2051

Master's Thesis

A Study on Discriminative Learning of Weighted Abduction

Kazeto Yamamoto

February 8, 2013

Graduate School of Information Science
Tohoku University

A Master's Thesis
submitted to Graduate School of Information Science,
Tohoku University
in partial fulfillment of the requirements for the degree of
MASTER of ENGINEERING

Kazeto Yamamoto

Thesis Committee:

Professor Kentaro Inui	(Supervisor)
Professor Kazuyuki Tanaka	(Co-supervisor)
Professor Tetsuo Kinoshita	(Co-supervisor)
Associate Professor Naoaki Okazaki	(Co-supervisor)

A Study on Discriminative Learning of Weighted Abduction*

Kazeto Yamamoto

Abstract

Abduction is inference to the best explanation. Abduction has long been studied in a wide range of contexts and is widely used for modeling artificial intelligence systems, such as diagnostic systems and plan recognition systems. Recent advances in the techniques of automatic world knowledge acquisition and inference technique warrant applying abduction with large knowledge bases to real-life problems. However, less attention has been paid to how to automatically learn score functions, which rank candidate explanations in order of their plausibility. In this paper, we propose a novel approach for learning the score function of first-order logic-based weighted abduction [1] in a supervised manner. Because the manual annotation of abductive explanations (i.e. a set of literals that explains observations) is a time-consuming task in many cases, we propose a framework to learn the score function from partially annotated abductive explanations (i.e. a subset of those literals). More specifically, we assume that we apply abduction to a specific task, where a subset of the best explanation is associated with output labels, and the rest are regarded as hidden variables. We then formulate the learning problem as a task of discriminative structured learning with hidden variables. Our experiments show that our framework successfully reduces the loss in each iteration on a plan recognition dataset.

Keywords:

*Master's Thesis, Department of System Information Sciences, Graduate School of Information Sciences, Tohoku University, B1IM2051, February 8, 2013.

Contents

List of Figures

List of Tables

1 Introduction

Making the implicit information (e.g. corefference relations, agent's plans, etc.) in sentences explicit is an important technique for various tasks in natural language processing. We are trying to construct discource understanding frameworks using abduction as the framework of making the implicit explicit.

Abduction is inference to the best explanation. Applying abduction to discource understanding was studied in 1980s and 1990s. The most important study of those is *Interpretation as Abduction* (IA) [1] by Hobbs et al. They showed the process of natural language interpretation can reasonably be described as abductive inference. For example, let us carry out interpretation on "John went to the bank. He got a loan." Treating world knowledge as background knowledge, target sentence to interpret as observation, IA applies abduction and outputs the best explanation as the result of interpretation. The interpretation of the example sentence with abduction is showed in Figure 1. From the result of interpretation, we can retrieve such implicit information as the goal of "went to the bank" is "got a loan" and that a coreference relation exists between "John" and "He".

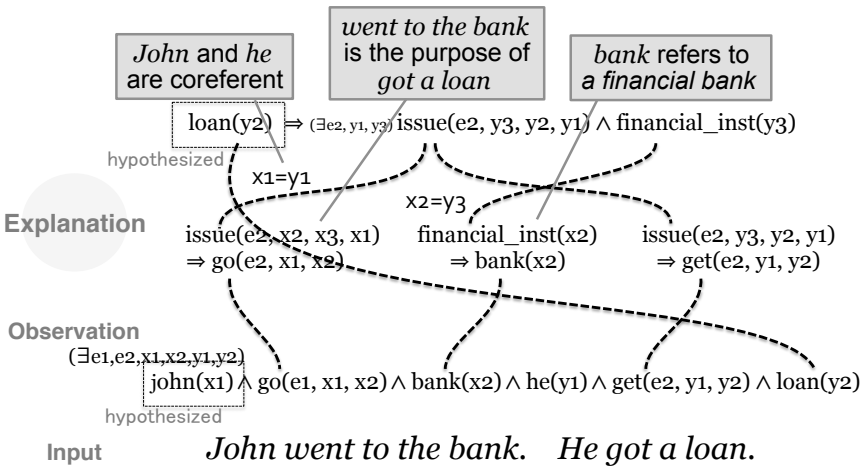

Figure 1: An example of discource interpretation with abduction

While the lack of world knowledge resources hampered applying abduction to real-life problems in the 1980s and 1990s, a number of techniques for acquiring world

knowledge resources have been developed in the last decade [2, 3, 4, 5, 6, etc.]. In addition, the development of an efficient inference technique of abduction warrant the application of abduction with large knowledge bases to real-life problems [7]. Consequently, several researchers have started applying abduction to real-life problems exploiting large knowledge bases. For instance, inspired by [1], [8] propose an abduction-based natural language processing framework using forty thousands axioms extracted from the popular ontological resources, WordNet [2] and FrameNet [3]. They evaluate their approach on the real-life natural language processing task of textual entailment recognition [9].

However, less attention has been paid to how to automatically learn score functions, which rank candidate explanations in order of their plausibility (henceforth, we call it the *cost function*). To apply abductive inference to a wide range of tasks, this non-trivial issue needs to be addressed because the criterion of plausibility is highly task-dependent, as discussed in Section 2.2. A notable exception is a series of studies [10, 11, 12], which emulate abduction in the probabilistic deductive inference framework, Markov Logic Networks (MLNs) [13]. MLN-based approaches can exploit several choices of weight learning methods originally developed for MLNs [14, 15, etc.]. However, MLN-based abduction has severe problems when they are applied to discource processing which we will discuss in Section 6.

In this paper, we propose a novel supervised approach for learning the cost function of first-order logic-based abduction. This is a framework to learn the cost function from subsets of explanations (henceforth, we call it *partial abductive explanations*). More specifically, we assume that we apply abduction to a specific task, where a subset of the best explanation is associated with output labels, and the rest are regarded as hidden variables. We then formulate the learning problem as the task of discriminative structured learning with hidden variables. As the cost function, we use the parametrized non-linear cost function proposed by [1].

This paper is organized as follows. We first give a brief review of abduction and the cost function of Hobbs et al.'s Weighted Abduction (Section 2). We then show our approach for learning the parametrized cost function of Weighted Abduction (Section 3). We start with the simple case where complete abductive explanations are given (Section 3.3), and then describe a method for learning with partial explanations (Section 3.4). Finally, we demonstrate how our learning algorithm works with a dataset of

plan recognition (Section 4).

2 Background

2.1 Abduction

Abduction is inference to the best explanation. Formally, logical abduction is defined as follows:

- **Given:** Background knowledge B, and observations O, where both B and O are sets of first-order logical formulas.
- **Find:** A *hypothesis* (*explanation*) H such that $H \cup B \models O$, $H \cup B \not\models \bot$, where H is a set of first-order logical formulas[1]. We say that p is *hypothesized* if $H \cup B \models p$, and that p is *explained* if $(\exists q)\ q \rightarrow p \in B$ and $H \cup B \models q$.

Typically, several hypotheses H explaining O exist. We call each of them a *candidate hypothesis*, and each literal in a hypothesis an *elemental hypothesis*. The goal of abduction is to find the best hypothesis among candidate hypotheses by a specific evaluation measure. In this paper, we formulate abduction as the task of finding the minimum-cost hypothesis $\hat{H}$ among a set $\mathcal{H}$ of candidate hypotheses. Formally, we find $\hat{H} = \arg\min_{H \in \mathcal{H}} c(H)$, where c is a function $\mathcal{H} \rightarrow \mathbb{R}$, which is called the *cost function*. We call the best hypothesis $\hat{H}$ the *solution hypothesis*. In the literature, several kinds of cost functions have been proposed, including cost-based and probability-based [16, 17, 1, 18, 12, etc.].

2.2 Weighted Abduction

Hobbs et al. [1] propose a cost function that can evaluate two types of plausibility of hypotheses simultaneously: *correctness* and *informativeness*. Correctness represents how reliable the contents of information are. Informativeness is how specific the information is. Hobbs et al. parametrized the cost function in a way that one can construct a cost function that favors more specific and thus more informative explanations, or

[1] Throughout the paper, $\models$ and $\bot$ represent logical entailment and logical contradiction respectively.

less specific but more reliable explanations in terms of a specific task by altering the parameters. The resulting framework is called *Weighted Abduction*.

In principle, the cost function gives a penalty for assuming specific and unreliable information but rewards for inferring the same information from different observations. To the best of our knowledge, Hobbs et al.'s Weighted Abduction is the only framework that considers the appropriateness of a hypothesis' specificity. Hobbs et al. exploit this cost function for text understanding where the key idea is that interpreting sentences is to find the lowest-cost abductive explanation[2] to the logical forms of the sentences in a agreement with a correctness-informativeness tradeoff. However, they do not elaborate on how to give the parameters of cost function. There are no existing methods for learning the parameters automatically.

Let us describe the cost function in more formal way. Following [1], we use the following representations for background knowledge, observations, and hypothesis throughout the paper:

- **Background knowledge** B: a set of first-order logical Horn clause whose literals in its body are assigned positive real-valued *weights*. We use a notation p^w to indicate "a literal p has the weight w" (e.g. $p(x)^{0.6} \wedge q(x)^{0.6} \Rightarrow r(x)$). We use a vector θ to represent weights on each literal, where each element refers to one weight on a literal in a specific axiom.

- **Observations** O: an existentially quantified conjunction of literals. Each literal has a positive real-valued cost. We use a notation $p^{\$c}$ to denote "a literal p has the cost c," and $c(p)$ to denote "the cost of the literal p" (e.g. $p(x)^{\$10} \wedge q(x)^{\$10}$).

- **Hypothesis** H: an existentially quantified conjunction of literals. Each literal also has a positive real-valued cost (e.g. $r(x)^{\$10} \wedge s(x)^{\$10}$). We define *unification* as an operation that merges two literals to one literal with the smaller cost, assuming that the arguments of the two literals are the same. For example, given the hypothesis $p(x)^{\$30} \wedge p(y)^{\$10} \wedge q(y)^{\$10}$, the unification of $p(x)^{\$30}$ and $p(y)^{\$10}$ yields another hypothesis $p(x)^{\$10} \wedge q(x)^{\$10}$, assuming $x = y$. The unification operation is applicable to any hypotheses, unless $H \cup B \not\models \bot$. We say that $p(x)$ is

[2]Hobbs did not mention cases where there are multiple lowest-cost explanations. So we assume those cases do not occur in this paper.

unified with $p(y)$ if $x = y \wedge c_\theta(p(x)) \geq c_\theta(p(y))$.[3]

Given a weight vector θ, the cost function of H is defined as the sum of all the costs of elemental hypotheses in H:

$$c_\theta(H) = \sum_{h \in P_H} c_\theta(h) \quad (1)$$

$$= \sum_{h \in P_H} \left[\prod_{i \in chain(h)} \theta_i \right] c(obs(h)), \quad (2)$$

where P_H is a set of elemental hypotheses that are not explained nor unified, $chain(h)$ is a set of indices to a literal in axioms that are used for hypothesizing h, and $obs(h)$ is an observed literal that is back-chained on to hypothesize h. Henceforth, we refer to a weight vector θ as the *parameter* of cost functions.

3 Discriminative Weight Learning of Weighted Abduction

In this section, we propose a method to learn the parameters of cost function in Weighted Abduction by recasting the parameter estimation problem as an online discriminative leaning problem with hidden variables.

The idea is four-fold:

1. We train the cost function with only partially specified gold abductive explanations which we represent as a partial set of the required literals (*gold partial explanations*).

2. We automatically infer complete correct abductive explanations from gold partial explanations by abductive inference.

3. We optimize the parameters of the cost function by minimizing the loss function where the loss is given by the difference of the costs of the minimal-cost hypothesis and the complete correct abductive explanations.

[3] If $cp(x) = cp(y)$, we regard that either $p(x)$ or $p(y)$ is unified.

4. We employ feed-forward neural networks to calculate the gradient of each parameter.

In the rest of this section, we first formalize explanation in Weighted Abduction with directed acyclic graphs (Section 3.1), and we then describe the outline of our learning method (Section 3.2) and elaborate on our learning framework in the simple case where complete abductive explanations are given (Section 3.3). We then describe a method for learning the parameters from partial abductive explanations (Section 3.4). Finally, we describe how to update the parameters through error back-propagation in FFNNs (Section 3.5).

3.1 Preliminaries

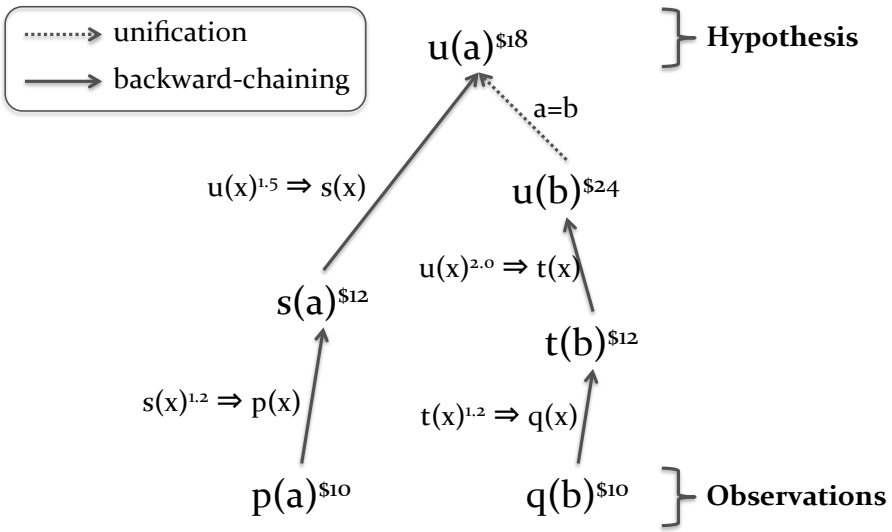

Figure 2: An example proof tree

In this paper, we express the hypotheses of Weighted Abduction as directed acyclic graphs (DAG). Namely, we regard each literal in the hypothesis as a node of DAG and each of relation between literals as an edge of DAG. We call these graphs *proof graph* and use a notation $G_{O,B,H}$ to denote the proof graph made from the observation O, the background knowledge B and the hypothesis H.

We define following two types of edge in proof graphs:

- **Backward-chaining:** Given the tail node's literal $p(x)^{\$c_1}$ and the head node's literal $q(x)^{\$c_2}$, this relation indicates that $q(x) \cup B \models p(x)$. Namely, $q(x)$ is hypothesized with $p(x)$. Then, the cost of head node's literal is caluclated by multiplication of the cost of tail node's literal and the weight of background knowledge (e.g. $c_2 = c_1 w$, where $q(x)^w \Rightarrow p(x)$).

- **Unification:** Given the tail node's literal $p(x)$ and the head node's literal $p(y)$, this relation indicate that $p(x)$ and $p(y)$ are unified and $x = y$.

Between the tail node and the head node of each edge in a proof graph, the relation that the head node's literal explain the tail node's literal exists. Thus, the set of literals of leaf nodes in proof graphs corresponds to P_H and the set of literals of root nodes in proof graphs corresponds to O.

We show an example proof graph in Figure 2. This is the proof graph made from the following background knowledge, observation and hypothesis:

$$\begin{aligned} B = \{ \quad & \forall x \quad (s(x)^{1.2} \Rightarrow p(x)), \forall x (s(x)^{1.2} \Rightarrow q(x)), \\ & \forall x \quad (u(x)^{1.5} \Rightarrow s(x)), \forall x (u(x)^{2.0} \Rightarrow t(x))\}, \qquad (3) \\ O = \quad & \exists x \quad (p(a)^{\$10} \wedge q(b)^{\$10}) \qquad (4) \\ H = \quad & \exists x \quad (u(a)^{\$18} \wedge u(b)^{\$24} \wedge s(a)^{\$12} \wedge t(b)^{\$12} \wedge a = b) \qquad (5) \end{aligned}$$

The cost of a hypothesis is calculated with Equation 1. Therefore, the cost of this hypothesis is calculated as $c(H) = \sum_{h \in P_H} c(h) = \$18$.

3.2 Outline of our method

In this section, we describe the outline of our learning method. The overall framework is illustrated in Figure 3.

First, we assume each training example to be a pair (O_i, τ_i), where O_i is an observation and τ_i is a gold partial explanation. A gold partial explanation is a set of literals that must be included in the correct abductive explanation T_i for the input observation O_i, i.e. $T_i \cup B \models O_i$ and $\tau_i \subseteq T_i$.

Next, we consider the online version of parameter learning [4]. For each cycle, given (O_i, τ_i), we perform Weighted Abduction for the observations O_i and background

[4]The batch version can also be considered by accumulating the gradients for each cycle before updating the weights.

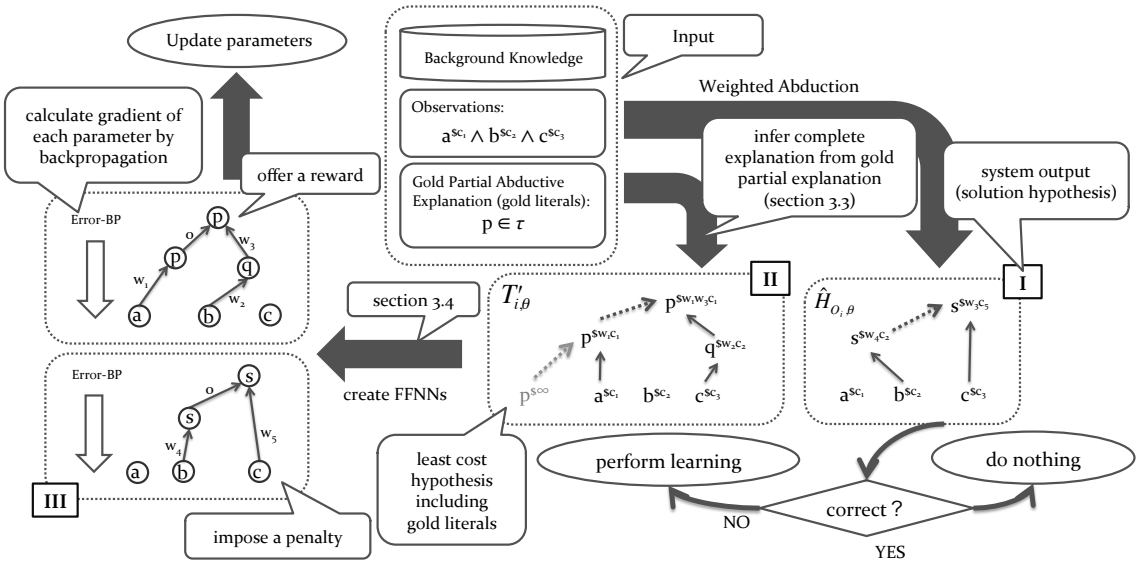

Figure 3: Outline of proposed parameter learning method

knowledge B with parameters θ, and get the solution hypothesis $\hat{H}_{O_i,\theta}$ ([I] in Figure 3). If $\hat{H}_{O_i,\theta}$ does not include τ_i (i.e. $\hat{H}_{O_i,\theta}$ is an incorrect prediction), we update the parameters so that $\hat{H}_{O_i,\theta}$ includes τ_i. In order to do so, we first infer a complete abductive explanation $\acute{T}_{i,\theta}$ from the gold partial explanation τ_i ([II]). We then update parameters θ by imposing a penalty to the wrong solution hypothesis $\hat{H}_{O_i,\theta}$ and offering a reward to the inferred correct complete abductive explanation $\acute{T}_{i,\theta}$. To compute these updates, we translate $\hat{H}_{O_i,\theta}$ and $\acute{T}_{i,\theta}$ to feed-forward neural networks and perform backpropagation on them ([III]).

In this paper, we assume that there is enough knowledge to infer the correct explanation in each problem (we call this *the knowledge completeness assumption*). If this assumption were not satisfied, which means that the correct explanation is not included in the candidate hypotheses, then we could not infer the correct explanation irrespectively of parameters. In the following discussion, we do not consider a case of knowledge base shortage.

3.3 Learning from complete abductive explanations

Let us first assume that we have a set of training examples labeled with a complete abductive explanation. Namely, we consider a training dataset $D = \{(O_1, T_1), (O_2, T_2), ..., (O_n, T_n)\}$, where O_i is an observation and T_i is the gold (correct) complete abductive explanation for O_i, i.e. $T_i \cup B \models O_i$.

For each labeled example (O_i, T_i), the solution hypothesis $\hat{H}_i$ is obtained by:

$$\hat{H}_i = \arg\min_{H \in \mathcal{H}_i} c_\theta(H) \tag{6}$$

We consider that a solution hypothesis $\hat{H}_i$ is correct if $\hat{H}_i = T_i$. Now we consider a loss function that calculates how far the current solution hypothesis is from the gold explanation, analogously to standard learning algorithms. If the current solution hypothesis is correct, the loss is zero. If $\hat{H}_i \neq T_i$, on the other hand, we consider the loss as given by the following loss function:

$$E(O_i, \theta, T_i) = \begin{cases} \frac{1}{2}\left(\frac{c_\theta(T_i) - c_\theta(\hat{H}_i)}{c_\theta(T_i) + c_\theta(\hat{H}_i)} + m\right)^2 + \lambda\theta \cdot \theta & (\hat{H}_i \neq T_i) \\ 0 & (\hat{H}_i = T_i) \end{cases}, \tag{7}$$

where $\lambda\theta \cdot \theta$ is a regularization term and m is a margin. Our goal is to learn the cost function c_θ that has minimal prediction errors. This goal is accomplished by learning parameters θ^* which minimize the total loss as below:

$$\theta^* = \arg\min_\theta \sum_{(O,T) \in D} E(O, \theta, T) \tag{8}$$

We describe how to minimize the loss in Section 3.5.

Note that we use the ratio of cost functions $\frac{1}{2}\left(\frac{c_\theta(T_i) - c_\theta(\hat{H}_i)}{c_\theta(T_i) + c_\theta(\hat{H}_i)} + m\right)^2$ as the loss function, instead of $\frac{1}{2}\left(c_\theta(T_i) - c_\theta(\hat{H}_i)\right)^2$. In the following, we shortly justify the use of the ratio of cost function. Let us suppose that we employ $\frac{1}{2}\left(c_\theta(T_i) - c_\theta(\hat{H}_i)\right)^2$ as the loss function. Then, we can minimize the loss function by minimizing the weight terms that appear in both $c_\theta(T_i)$ and $c_\theta(H_i)$, namely the weights assigned to axioms that are used in both T_i and H_i. For instance, given $O_i = \{p(a)^{\$c}\}$, $B = \{q(x)^{w_0} \Rightarrow p(x), s(x)^{w_1} \Rightarrow q(x), t(x)^{w_2} \Rightarrow q(x)\}$, $T_i = \{s(a)^{\$w_0 w_1 c}\}$, $H_i = \{t(a)^{\$w_0 w_2 c}\}$, we can minimize the value of loss function by minimizing the value of w_0. As a result, the learning procedure just decreases w_0 as much as possible to minimize the loss function. This prevents our framework from

learning a meaningful cost function, because the minimization of weights does not imply that we can infer the gold hypothesis as the solution hypothesis. To avoid this problem, we employ the ratio of cost functions.

3.4 Learning from partial abductive explanations

In the above, we assumed that each training example has a complete abductive explanation. However, this assumption is not realistic in many cases because it is usually prohibitively costly for human annotators to give a complete abductive explanation for each given input. This leads us to consider representing a training example as a pair of observation O_i and gold partial explanation τ_i, which is a partial set of literals that must be included in the explanation of O_i. In the case of Figure 3, we assumed that the correct hypothesis for the given observation is partially specified by the literal $p \in \tau$.

This way of simplification is essential in real-life tasks. In plan recognition, for example, it is not an easy job for human annotators to give a complete explanation to an input sequence of observed events, but they can tell whether it is a shopping story or a robbing story much more easily, which can be indicated by a small set of gold literals.

Now, our goal is to learn the cost function from partial explanations $D = \{(O_1, \tau_1), (O_2, \tau_2), ..., (O_n, \tau_n)\}$. Regarding whether each gold literal is included in the solution hypothesis $\hat{H}_{O_i,\theta}$ and the structure of the proof graph $G_{O_i,B,\hat{H}_{O_i,\theta}}$ as hidden states, this task can be seen as discriminative structure learning with hidden states. The issue is how to infer the complete correct explanation $\acute{T}_{i,\theta}$ from a given incomplete set τ_i of gold literals. Fortunately, this can be done straightforwardly by adding the gold literals τ_i to the observation O_i:

$$O_i^+ = O_i \cup \{t^{\$\infty} \mid t \in \tau_i\}, \tag{9}$$

where each gold literal is assigned an infinitive cost. Then, the solution hypothesis $\hat{H}_{O_i^+,\theta}$ is equivalent to the complete correct explanation $\acute{T}_{i,\theta}$ if the following conditions are satisfied:

- A hypothesis including τ_i exists in the candidate hypotheses for O_i (the knowledge completeness assumption).
- $\hat{H}_{O_i^+,\theta}$ has no backward chaining from $t \in \tau_i$.

Figure 3 ([II]) illustrates a simple case, where $\hat{H}_{O_i^+,\theta}$ is inferred by adding the gold literal p to the observation. Since this added literal p is assigned an infinitive cost, it is strongly motivated to derive an explanation including that p, resulting in obtaining the correct explanation $\acute{T}_{i,\theta}$.

When these conditions are satisfied, because each t has a huge cost, the system selects as the solution hypothesis $\hat{H}_{O_i^+,\theta}$ the hypothesis in which most literals in τ_i unify with other literals. Then, assuming the existence of a hypothesis including τ_i in the candidate hypotheses for O_i, there is the hypothesis in which each of the literals in τ_i unifies to a literal in the candidate hypotheses for O_i^+, and it is selected as solution hypothesis $\hat{H}_{O_i^+,\theta}$. Because the cost of t must be 0 when it is unified with an other literal included in O_i or hypothesized from O_i, the cost of $\hat{H}_{O_i^+,\theta}$ is equal to cost of $\acute{T}_{i,\theta}$. So $\hat{H}_{O_i^+,\theta}$ must be equal to $\acute{T}_{i,\theta}$.

It should be note that we can check whether candidate hypotheses satisfy the above-mentioned conditions by checking the cost of the solution hypothesis, because any non-unified $t^\$\infty$ will result in a huge cost.

3.5 Updating parameters with FFNNs

To update parameters, we want to compute the gradient of the loss function for each parameter. However, since the cost function and the loss function are both nonlinear to their parameters, their gradients cannot be computed straightforwardly.

To solve this problem, we propose employing feed-forward neural networks (FFNNs). An FFNN is a directed acyclic graph where the output of each node j is given by:

$$z_j = h(a_j), \tag{10}$$

$$a_j = \sum_{i \in \{i | e_{i \to j} \in \mathbb{E}\}} z_i \times \mathbf{w}_{i \to j}, \tag{11}$$

where z_i denotes the output of node i, a_i denotes the degree of activation of node i, $h(a)$ is an activation function, $e_{i \to j}$ denotes a directed edge from node i to node j, and $\mathbf{w}_{i \to j}$ denotes the weight of $e_{i \to j}$.

Then, we express the cost function of H with a FFNN. This is achieved by applying the following convertion to $G_{O,B,H}$:

1. The cost of each literal in $G_{O,B,H}$ is the output of the node in the corresponding FFNN.

2. Each backward-chaining edge in $G_{O,B,H}$ is an edge with weight w in the FFNN where w denote the weight of the background knowledge of the corresponding backward-chaining edge.

3. Each unification edge in $G_{O,B,H}$ is an edge with weight 0 in the FFNN.

4. The activation function of each layer in FFNNs is $h(a) = a$.

5. An *output node*) is added to the FFNN, making new edges with weight 1 between output node and each node that corresponds to each literal in P_H (i.e. leef nodes in the proof graph).

Then, the value of the output node is equal to the cost function $c_\theta(H)$ in Weighted Abduction.

We show that the cost function of Weighted Abduction is converted into equivalent FFNNs as shown in Figure 4. This indicates the FFNN can express the cost function of Weighted Abduction. Therefore, we are able to apply various techniques in FFNNs to learning parameters of Weighted Abduction. Namely, gradients of the loss function can be caluclated easily by using the backpropagation technique of FFNNs.

Moreover, FFNNs are flexible framework and can express various functions by changing the activation functions or the network's structure. Thus, this idea can be apply to not only Weighted Abduction but other various frameworks of abduction.

3.6 Procedures of parameter learning

The overall learning procedure is given in Algorithm 1. First, the solution hypothesis is inferred from observation O, and if it does not include gold literals τ, it is treated as a negative example H^- (Line 3-6). Next, the positive example H^+ is inferred from observation O^+ (Line 7,8). The loss is then calculated from the costs of H^+ and H^- (Line 9) H^+ and H^- is converted into FFNNs (Line 10). The gradient of the loss function for each non-zero cost literal is assigned to the corresponding node in the FFNN (Line 11-15). The gradients of the loss function for costs of the other literals are calculated by applying standard backpropagation to the converted FFNNs (Line 16). Updating the parameters is performed with these gradients (Line 17). The parameters are trained iteratively until the learning converges.

Algorithm 1 parameter learning

Input: $B, \theta, \mathbb{D}$
repeat
  for all $(O, \tau) \in \mathbb{D}$ **do**
    $\hat{H} \leftarrow Inference(O, \theta)$
    if $\tau \not\subseteq \hat{H}$ **then**
      $H^- \leftarrow \hat{H}$
      $O^+ = O \cup \{t^{\$\infty} \mid t \in \tau\}$
      $H^+ \leftarrow Inference(O^+, \theta)$
      $E_{O,\theta} \leftarrow LossFunction(H^+, H^-)$
      $N \leftarrow MakeFFNN(H^+, H^-)$
      for all $h \in P_{H^+} \cup P_{H^-}$ **do**
        if $c(h) > 0$ **then**
          assign gradient $\frac{\partial E_{O,\theta}}{\partial c(h)}$
        end if
      end for
      do backpropagation
      $\theta \leftarrow UpdateWeights(\theta, \nabla E_{O,\theta})$
    end if
  end for
until convergence
Output: θ

4 Featurizing parameters

So far, we have assigned a parameter for each literal which corresponds to a particular background knowledge. However, in this setting, we can train parameters for the background knowledge only appear in the training data, therefore the trained system would not be able to deal with unseen data. In this section, we describe a method which featurizes parameters of weighted abduction and its learning algorithm.

We introduce a function which defines parameter values:

$$\theta_i = h(\mathbf{F}_i \cdot \phi) \tag{12}$$

where $\mathbf{F}_i \in \mathbb{R}^n$ is the feature vector for a parameter θ_i which corresponds to a particular

background knowledge, $\phi \in \mathbb{R}^n$ is the weights for features and $h(\cdot)$ is the activation function. In training, instead of parameters, feature weights ϕ are trained based on a particular loss function. In Weighted Abduction, since parameter values must be more than 1.0, we employ the following activation function:

$$h(a) = \frac{1.0}{n} + exp(a) \tag{13}$$

where n is the number of literals in the left hand side of the background knowledge which the literal belongs to. Gradient values of feature weights can be calculated from gradient values of parameters as follows:

$$\frac{\partial E_{O,\theta}}{\partial \phi_k} = \sum_i \left(\frac{\partial E_{O,\theta}}{\partial \theta_i} \mathbf{F}_{ik} \right) \tag{14}$$

where ϕ_k is k^{th} element of the feature weights, $\mathbf{F}_{ik}$ is k^{th} element of the feature vector for a parameter θ_i.

5 Evaluation

5.1 Evaluation for ablity to learn parameters

We evaluate the proposed learning procedure on the dataset of plan recognition. In this experiment, we address the following questions: (i) does our leaning procedure actually decrease prediction errors? (ii) are models trained by our learning procedure robust to unseen data? To answer these questions, we evaluate prediction performance on a plan recognition dataset in the two settings: a closed test (i.e., the same dataset is used for both training and testing) and an open test (i.e., two distinct datasets are used for training and testing). In order to obtain the lowest-cost hypotheses, we used the Integer Linear Programming-based abductive reasoner proposed by [7].

5.1.1 Dataset

We used [19]'s story understanding dataset, which is widely used for evaluation of abductive plan recognition systems [10, 18, 12]. In this dataset, we need to abductively infer the top-level plans of characters from actions which are represented by the logical forms. For example, given "*Bill went to the liquor-store. He pointed a gun at*

the owner," plan recognition systems need to infer *Bill*'s plan. The dataset consists of development set and a test set, each of which includes 25 plan recognition problems. The dataset contains on average 12.6 literals in observed logical forms. The background knowledge base contains of 107 Horn clauses. Figure 5 shows an example of this dataset.

In our evaluation, we introduced two types of axioms in addition to the original 107 axioms. First, to make the predicates representing top-level plans (e.g. shopping, robbing) disjoint, we generated 73 disjointness axioms (e.g. $robbing(x) \Rightarrow \neg shopping(x)$). Note that it is still possible to infer multiple top-level plans for one problem, because we are able to hypothesize $robbing(x) \wedge shopping(y)$. Second, we generated axioms of superplan-subplans relations (e.g. $going\_by\_plane(x) \Rightarrow going\_by\_vehicle(x)$). In total, we used 220 background axioms for our evaluation.

For evaluating the prediction performance of our system, we focused on how well the system infers top-level plans, and their subparts (i.e. subplans, role-fillers), following [12]. More specifically, we use precision (ratio of inferred literals that are correct), recall (ratio of correct literals that are inferred by the system), and F-measure (harmonic mean of precision and recall), because the gold data often has multiple top-level plan predicates.

5.1.2 Experimental setting

We applied weight regularization in order to prevent overfitting to the training set. The hyperparameter for regularization λ was set to 0.1. For parameter updating, we employed the annealing approach; $\theta_{new} = \theta - \eta_0 k^i \nabla E_\theta$ where η_0 (initial learning rate) was set to 0.0001, k (annealing parameter) was set to 0.95 and i is the number of iterations. The hyperparameters were selected based on performances on the development set. All weights were initialized to 0.0.

5.1.3 Results and discussion

At first, we report results of the closed test where the development set was used for both training and testing. Figure 6 shows the values of the loss function at each iteration on the development set. The curve indicates that our learning procedure successfully reduces values of the loss function at each iteration. The reason for the fluctuation in values is thought to be the existence of hidden variables.

In the open test, we trained our model on the development set and then tested on the test set. Figure 7 shows plots of values of the three measures (i.e. Precision, Recall and F-measure) on the test set at each iteration. Although the values are also fluctuate as with the closed test, performance rises in terms of all measures compared to the performances at iteration zero (i.e. initial values). The results suggest that the learning procedure is robust to unseen data.

Singla and Mooney [12] report that the MLN-based approach achieve 72.10 F-measure on the same test set, which is slightly better than our results. However, our experimental setting and Singla's are different on various point such as framework of abduction (i.e. Weighted Abduction vs. MLN-based abduction), method of parameter learning (i.e. FFNNs vs. MLNs), method of parameter initialization (i.e. constant value vs. manually tuning). Therefore, it is unable to compare usefulness of these frameworks.

It has taken about half an hour to perform training for each iteration. Most of the time was spent in obtaining solution hypotheses using ILP-based abductive inference.

5.2 Evaluation for featurizing

We evaluate how effective is featurizing parameters in Weighted Abduction. In this experiment, we evaluate prediction performance on a plan recognition dataset in 10-fold cross validation on some feature settings. The abductive reasoner and the hyperparameters are same as in the evaluation in Section 5.1.

5.2.1 Features

We evaluated three settings on featurizing the paramters. Table 1 shows the details about the features used in the experiment and Table 2 shows three settings we used in the experiment. LITERAL and PRED features include information of individual literals and predicates included in axioms. CLAUSE features capture what axioms are used in a hypothesis. CLAUSE-OBS and PRED-OBS features combine predicate or clause with observation information. The reason for introducing CLAUSE-OBS and PRED-OBS is to capture dependence between parameter weights of Weighted Abduction and observations. Setting I corresponds original Weighted Abduction. Setting II is more generalized than Setting I. In addition, Setting III considers dependency between pa-

Feature	Explanation	Example feature value
LITERAL	each literal included in the left-hand side	$go\_step(s, g)$, $inst\_shopping(s)$
PRED	each predicate included in the left-hand side	go_step, $inst\_shopping$
CLAUSE	whole clause of axiom	$inst\_shopping(s) \wedge go\_step(s, g)$ $\rightarrow inst\_going(g)$
CLAUSE-OBS	combination of CLAUSE and each predicate in observations	$inst\_shopping(s) \wedge go\_step(s, g)$ $\rightarrow inst\_going(g)$ & $inst\_robbing$
PRED-OBS	combination of PRED and each predicate in observations	go_step & $inst\_robbing$

Table 1: Features used for the system. We show example feature values for the axioms $inst\_shopping(s) \wedge go\_step(s, g) \rightarrow inst\_going(g)$, and the observation $inst\_robbing(R)$.

Feature	Setting I	Setting II	Setting III
LITERAL	✓	✓	✓
PRED		✓	✓
CLAUSE		✓	✓
CLAUSE-OBS			✓
PRED-OBS			✓

Table 2: Settings of features used in the experiment.

rameter weights of Weighted Abduction and observations.

5.2.2 Results and discussion

Figure 8 shows the results of the experiment. The result indicates that our extention about the parameters successfully improves robustness to unseen data and correct assignment of the weight of axioms depend on the observations.

6 Related work

As mentioned in Section 1, abduction has been extensively studied in a wide range of contexts. However, less attention has been paid to how to automatically learn cost functions. In the field of Statistical Relational Learning, some researchers [10, 12, 11,

etc.] employ Markov Logic Networks [13] to emulate abductive inference. MLNs provide well-studied software packages of inference and learning.

However, MLN-based approaches require special procedures to convert abduction problems into deduction problems because of the deductive nature of MLNs. The pioneering work of MLN-based abduction [10] converts background axioms into MLN logical formulae by (i) reversing implication and (ii) constructing axioms representing mutual exclusiveness of explanation (e.g. the set of background knowledge axioms $\{p_1 \rightarrow q, p_2 \rightarrow q,\ p_3 \rightarrow q\}$ is converted into the following MLN formulae: $q \rightarrow p_1 \vee p_2 \vee p_3$, $q \rightarrow \neg p_1 \vee \neg p_2$, $q \rightarrow \neg p_1 \vee \neg p_3$ etc.). As the readers can imagine, MLN-based approach suffers from the inefficiency of inference due to the increase of converted axioms. Therefore, learning would not scale to larger problems due to the severe overhead [20]. [12] report that their MLN-based abduction models cannot be trained in larger dataset.

Moreover, when MLN-based approaches are applied to abduction-based discourse processing, a critical problem arises. MLN-based approaches represent a hypothesis as a truth assignment to ground atoms in the Herbrand base of background knowledge, while our framework represents a hypothesis as a set of first-order literals or equalities of logical variables. This means that a hypothesis generated by MLN-based approaches loses the first-order information in the input text. As shown in Section 1, each logical variable in the observation corresponds to a mention in the discourse; thus losing this information would be a serious drawback in discourse processing. For example, suppose that MLN-based approaches produce the hypothesis $president(A)$, $male(A)$, $doctor(B)$, $male(B)$ (A and B are constants) to the observation $\exists p, m_1, d, m_2\{president(p) \wedge male(m_1) \wedge doctor(d) \wedge male(m_2)\}$. Then, we can interpret this hypothesis as two types of first-order logical forms: $president(p) \wedge male(m_1) \wedge doctor(d) \wedge male(m_2) \wedge p = m_1 \wedge d = m_2$, or $president(p) \wedge male(m_1) \wedge doctor(d) \wedge male(m_2) \wedge p = m_2 \wedge d = m_1$. This means that we cannot decide which discourse mentions are identified as coreferential in the hypothesis generated by MLN-based approaches. Some previous work [21, 22] represent coreference relations by introducing special predicates that describe two logical variables are equal, but they use MLNs to create a classifier (i.e. binary log-linear classification model that utilizes a number of features) rather than reasoner. Therefore, it is a non-trivial issue to use these coreference representations with logical inference aimed at complicated commonsense

reasoning, which is our goal in abudction-based discourse processing.

7 Conclusion

We have proposed a supervised approach for learning the cost function of Weighted Abduction. We formulated the learning procedure in the framework of structured learning with hidden variables. Our approach enables us to learn the non-linear cost function from partial abductive explanations, which is the typical situation in real-life tasks because constructing complete abductive explanations is usually a cost-consuming task. To the best of our knowledge, this is the first work to address the issue of automatic parameter learning of the cost function of Weighted Abduction, which can evaluate both the correctness and informativeness of explanations. In our evaluation, we found that our learning procedure can reduce the value of loss function in each iteration, and learned weights are also robust to unseen dataset.

Our future work includes large-scale evaluation of our learning procedure. We plan to evaluate our procedure on the popular natural language processing tasks, coference resolution with a massive set of axioms extracted from several language resources (e.g. WordNet [2]). It is also a problem that it takes long time to training weights. This problem will be critical in training on a large data set. We will address this problem by improving of abductive reasoner and optimization methods. As discussed in [1], coreference relation correponds to the unification of two logical variables. We therefore plan to incorporate a term that represents the cost of variable unification in the cost function of Weighted Abduction.

Acknowledgements

This section is written in Japanese...

本研究を進めるにあたり，ご指導を頂いた乾健太郎教授，岡崎直観准教授，邉陽太郎助教に感謝致します．

また，日常の議論を通じて多くの知識や示唆を頂いた乾・岡崎研究室の皆様に感謝致します．特に井之上直也氏には，研究に関する議論や推論エンジンの提供．論文執筆のサポートなど，修士課程の 2 年間を通じて数えきれないほど助けて頂きました．本当にありがとうございました．

陰ながら私を支えてくださった父と母に感謝致します．苦しい経済状況にも関わらず常に私の事を第一に考えてくれるその姿は，私にとって，精神的にも大きな支えでした．ありがとうございました．

最後に，東北大学，そして職員の皆さまに感謝を述べたいと思います．ありがとうございました．

References

[1] J. R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. *Artificial Intelligence*, Vol. 63, pp. 69–142, 1993.

[2] C. Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

[3] J. Ruppenhofer, M. Ellsworth, M.R. Petruck, C.R. Johnson, and J. Scheffczyk. FrameNet II: Extended Theory and Practice. Technical report, Berkeley, USA, 2010.

[4] N. Chambers and D. Jurafsky. Unsupervised Learning of Narrative Schemas and their Participants. In *ACL*, pp. 602–610, 2009.

[5] S. Schoenmackers, J. Davis, O. Etzioni, and D. Weld. Learning First-order Horn Clauses from Web Text. In *EMNLP*, pp. 1088–1098, 2010.

[6] D. Hovy, C. Zhang, E. Hovy, and A. Penas. Unsupervised discovery of domain-specific knowledge from text. In *ACL*, pp. 1466–1475, 2011.

[7] N. Inoue and K. Inui. ILP-Based Reasoning for Weighted Abduction. In *AAAI Workshop on Plan, Activity and Intent Recognition*, 2011.

[8] E. Ovchinnikova, N. Montazeri, T. Alexandrov, J. R Hobbs, M. McCord, and R. Mulkar-Mehta. Abductive Reasoning with a Large Knowledge Base for Discourse Processing. In *IWCS*, pp. 225–234, Oxford, UK, 2011.

[9] I. Dagan, B. Dolan, B. Magnini, and D. Roth. Recognizing textual entailment: Rational, evaluation and approaches - Erratum. *NLE*, Vol. 16, No. 1, p. 105, 2010.

[10] R. J. Kate and R. J. Mooney. Probabilistic Abduction using Markov Logic Networks. In *PAIRS*, 2009.

[11] J. Blythe, J. R. Hobbs, P. Domingos, R. J. Kate, and R. J. Mooney. Implementing Weighted Abduction in Markov Logic. In *IWCS*, pp. 55–64, Oxford, UK, 2011.

[12] P. Singla and P. Domingos. Abductive Markov Logic for Plan Recognition. In *AAAI*, pp. 1069–1075, 2011.

[13] M. Richardson and P. Domingos. Markov logic networks. *ML*, pp. 107–136, 2006.

[14] T. N. Huynh and R. J. Mooney. Max-Margin Weight Learning for Markov Logic Networks. In *Proceedings of the International Workshop on Statistical Relational Learning (SRL-09)*, 2009.

[15] D. Lowd and P. Domingos. Efficient Weight Learning for Markov Logic Networks. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD 2007, pp. 200–211, 2007.

[16] E. Charniak and R. P. Goldman. A Probabilistic Model of Plan Recognition. In *AAAI*, pp. 160–165, 1991.

[17] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, Vol. 64 (1), pp. 81–129, 1993.

[18] S. Raghavan and R. J. Mooney. Bayesian Abductive Logic Programs. In *STARAI*, pp. 82–87, 2010.

[19] H. T. Ng and R. J. Mooney. Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation. In *KR*, pp. 499–508, 1992.

[20] N. Inoue and K. Inui. Large-scale Cost-based Abduction in Full-fledged First-order Logic with Cutting Plane Inference. In *Proceedings of the 12th European Conference on Logics in Artificial Intelligence*, p. to appear, 2012.

[21] Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with markov logic. In *Proceedings of EMNLP*, pp. 650–659, 2008.

[22] Yang Song, Jing Jiang, Wayne Xin Zhao, Sujian Li, and Houfeng Wang. Joint learning for coreference resolution with markov logic. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1245–1254. ACL, July 2012.

List of Publications

Awards

- NLP 若手の会 第 7 回シンポジウム 奨励賞
- 第 206 回自然言語処理研究会 学生奨励賞

International Conferences Papers

- Kazeto Yamamoto, Naoya Inoue, Yotaro Watanabe, Naoaki Okazaki and Kentaro Inui. Discriminative Learning of First-order Weighted Abduction from Partial Discource Explanations. In Proceedings of the 14th International Conference on Intelligent Text Processing and Computational Linguistics. March 2013.

Other Publications

- 山本風人, 井之上直也, 渡邉陽太郎, 岡崎直観, 乾健太郎. 重み付き仮説推論における部分的な正解仮説からの識別学習. NLP 若手の会 第 7 回シンポジウム. September 2012.
- 山本風人, 井之上直也, 渡邉陽太郎, 岡崎直観, 乾健太郎. 誤差逆伝播を利用した重み付き仮説推論の教師有り学習. 情報処理学会研究報告, Vol.2012-NL-206. May 2012.

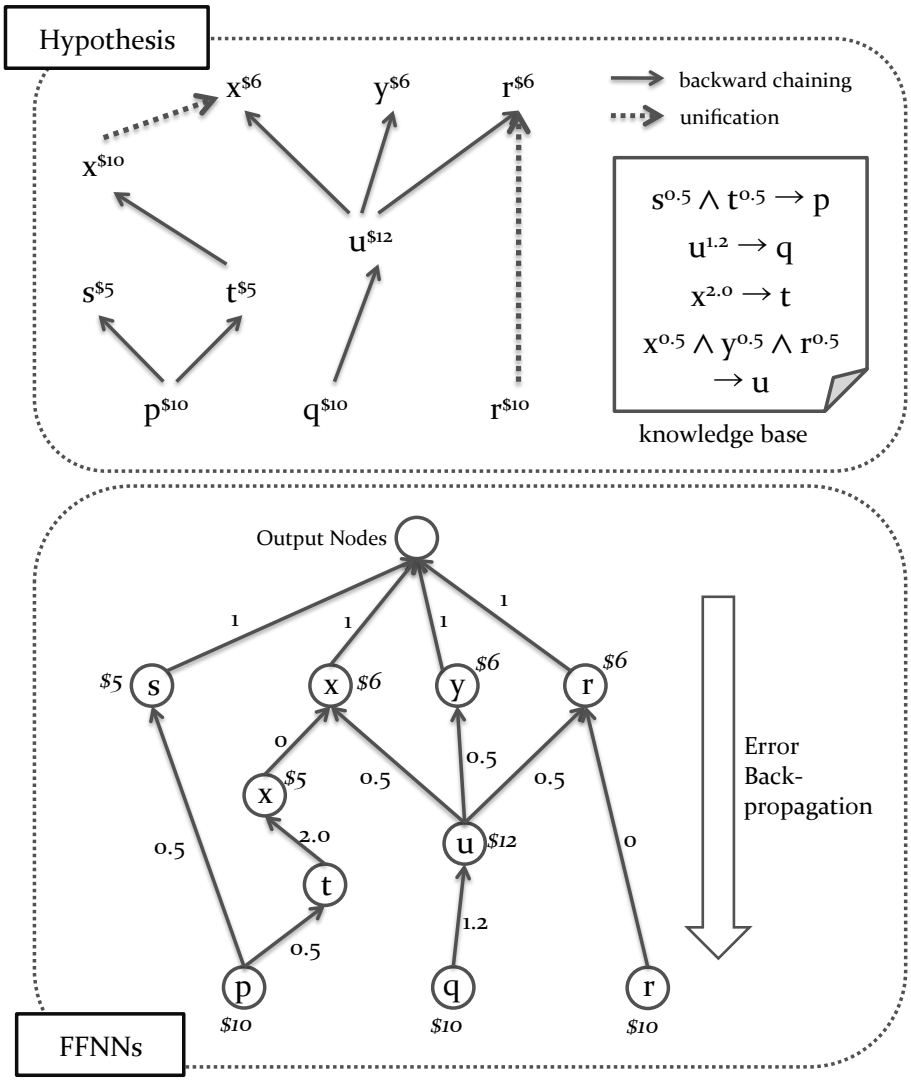

Figure 4: Example of transforming hypotheses into FFNNs

(a) Observations

"Bill went to the store. He paid for some milk"

$instance\_going(GO1)$

$goer(GO1, BILL)$

$destination\_go(GO1, STORE)$

$instance\_paying(PAY1)$

$payer(PAY1, BILL)$

$thing\_paid(PAY1, MILK)$

(b) Correct abductive explanations

$instance\_shopping(s)$

$shopper(s, BILL)$

$go\_step(s, GO1)$

$pay\_step(s, PAY1)$

$thing\_shopped\_for(s, MILK)$

(c) Background knowledge

$instance\_shopping(s) \wedge go\_step(s, g) \rightarrow instance\_going(g)$

$instance\_shopping(s) \wedge pay\_step(s, pay) \rightarrow instance\_paying(pay)$

$instance\_shopping(s) \wedge go\_step(s, g) \wedge shopper(s, p) \rightarrow goer(g, p)$

$instance\_shopping(s) \wedge pay\_step(s, pay) \rightarrow payer(pay, p)$

$instance\_shopping(s) \wedge go\_step(s, g) \wedge store(s, str) \rightarrow destination\_go(g, str)$

$instance\_shopping(s) \wedge pay\_step(s, pay) \wedge thing\_shopped\_for(s, t) \rightarrow thing\_paid(pay, t)$

Figure 5: Example dataset.

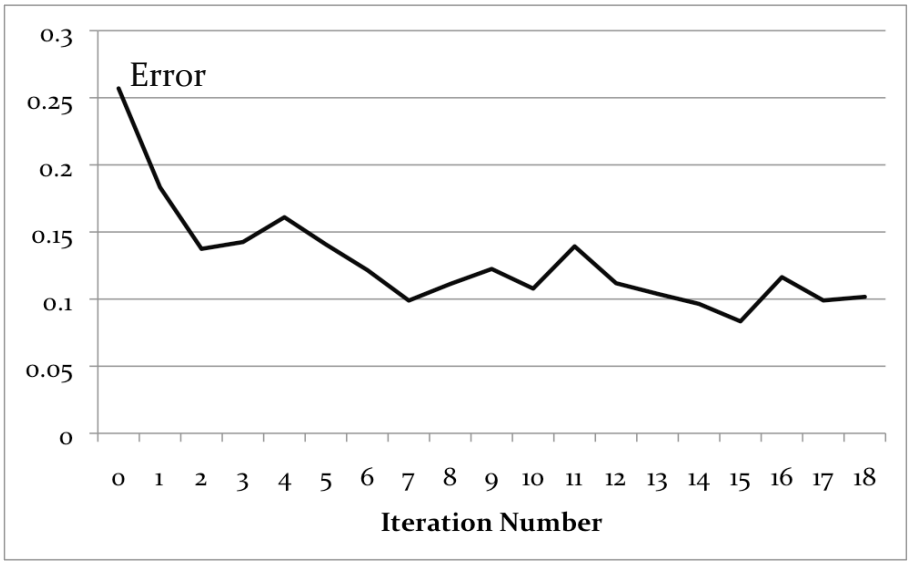

Figure 6: Loss function values (closed test)

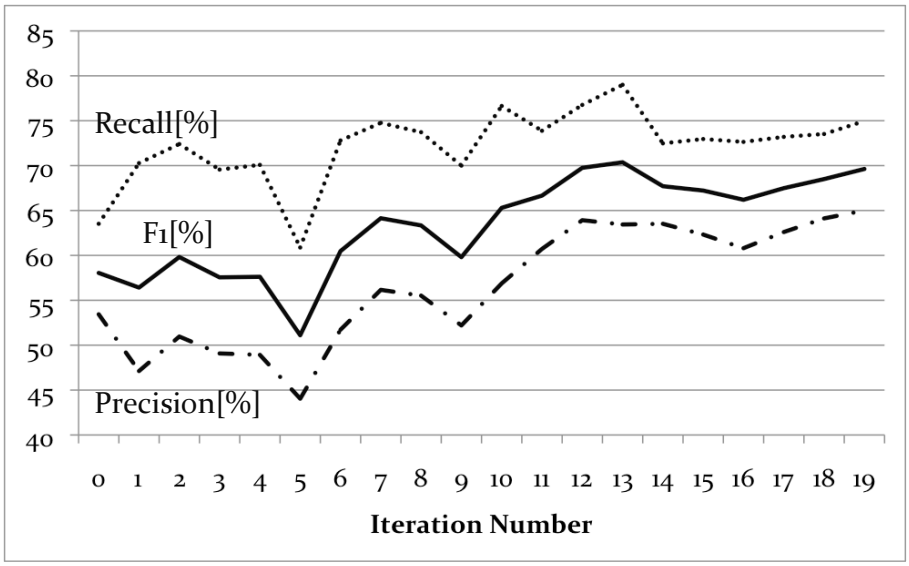

Figure 7: Open test results.

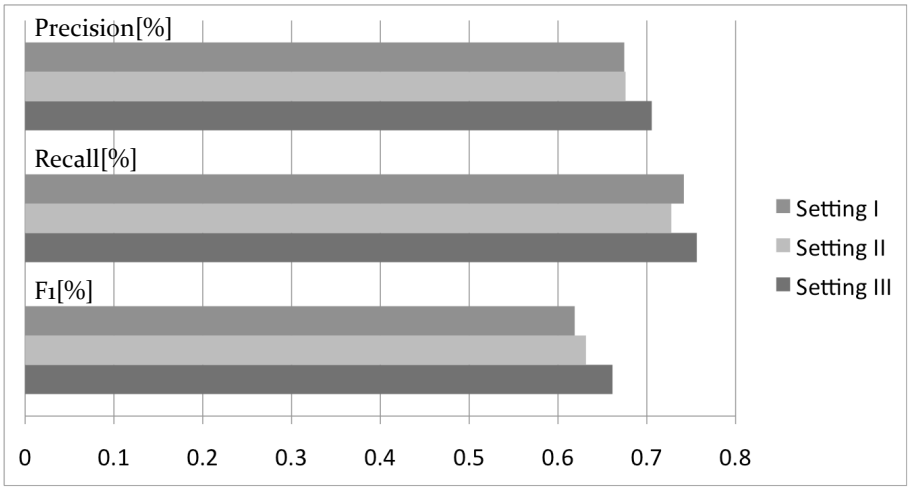

Figure 8: Results on each feature setting.