Learning Dependencies between Case Frame Slots

Hang Li*
Real World Computing Partnership

Naoki Abe*
Real World Computing Partnership

A theoretically sound method for learning dependencies between case frame slots is proposed. In particular, the problem is viewed as that of estimating a probability distribution over the case slots represented by a dependency graph (a dependency forest). Experimental results indicate that the proposed method can bring about a small improvement in disambiguation, but the results are largely consistent with the assumption often made in practice that case slots are mutually independent, at least when the data size is at the level that is currently available.

1. Introduction

We address the problem of automatically acquiring case frame patterns (or their near equivalents, selectional patterns and subcategorization patterns) from large corpus data. In our view, the acquisition of case frame patterns involves the following three subtasks: (1) extracting case frame instances from corpus data, (2) generalizing case frame slots within the case frames, and (3) learning dependencies that exist between the (generalized) case frame slots. We consider here the third of these subtasks and propose a method of learning dependencies between case frame slots.

The term "dependency" refers to the relationship that may exist between case slots and that indicates strong co-occurrence between the values of those case slots. For example, consider the following sentences:

(1) She flies jets.

(2) That airline company flies jets.

(3) She flies Japan Airlines.

(4) *That airline company flies Japan Airlines.

We see that *airline company* can be the value of the arg1 (subject) slot, when the value of the arg2 (direct object) slot is *airplane* but not when it is *airline company*. These sentences indicate that the possible values of case slots depend in general on those of others: dependencies between case slots exist.[1]

The knowledge of dependencies between case slots is useful in various tasks in natural language processing, especially in analyzing sentences involving multiple prepositional phrases, such as *The girl will fly a jet from Tokyo to Beijing*. Note in this example

* c/o C&C Media Res. Labs. NEC, 4-1-1 Miyazaki Miyamae-ku, Kawasaki, 216-8555 Japan. E-mail: {lihang,abe}@ccm.cl.nec.co.jp

1 One may argue that these examples involve different word senses of *fly*, and in general, that if word senses were disambiguated there would be no dependency between case slots. But, with that interpretation, word senses would have to be automatically disambiguated given the corpus data, and we would find ourselves left with the same problem. Furthermore, word senses are in general difficult to define precisely, and we feel it is better not to rely on this notion in a natural language application, unless it is necessary.

Table 1
Example case frames generated by a class-based model.

Case Frame	Frequency
(fly (arg1 ⟨person⟩)(arg2 ⟨airplane⟩))	3
(fly (arg1 ⟨company⟩)(arg2 ⟨airplane⟩))	2
(fly (arg1 ⟨person⟩)(arg2 ⟨company⟩))	1
(fly (arg1 ⟨person⟩)(to ⟨place⟩))	1
(fly (arg1 ⟨person⟩)(from ⟨place⟩)(to ⟨place⟩))	1
(fly (arg1 ⟨company⟩)(from ⟨place⟩)(to ⟨place⟩))	1

that the slot of *from* and that of *to* should be considered dependent and the attachment site of one of the prepositional phrases (case slots) can be determined by that of the other with high accuracy and confidence. There has not been a general method proposed to date, however, that learns dependencies between case slots. Methods of resolving ambiguities have been based, for example, on the assumption that case slots are mutually independent (Hindle and Rooth 1991), or at most two case slots are dependent (Collins and Brooks 1995). In this article, we propose an efficient and general method of learning dependencies between case frame slots.

2. Learning Method

Suppose that we have frequency data of the type shown in Table 1 automatically extracted from a corpus, in which words in the slots are replaced by the classes they belong to. We assume that case frame instances with a given verb are generated by a discrete joint probability distribution of the form

$$P_Y(X_1, X_2, \ldots, X_n),$$

where P_Y stands for the verb, and each of the random variables $X_i, i = 1, 2, \ldots, n$, represents a case slot. We then formulate the dependencies between case slots as the *probabilistic* dependencies between these random variables.

Such a joint distribution can be represented by three alternative types of probabilistic models according to the type of values each random variable X_i assumes. When X_i assumes a word or a special symbol "0" as its value, we refer to the corresponding model as a **word-based model**. Here 0 indicates the absence of the case slot in question. When X_i assumes a word-class (such as ⟨person⟩ or ⟨company⟩) or 0 as its value, the corresponding model is called a **class-based model**. When X_i takes on 1 or 0 as its value, we call the model a **slot-based model**. Here the value of 1 indicates the presence of the case slot in question, and 0 the absence thereof.

The number of parameters in a joint distribution will be exponential, however, if we allow interdependencies among all of the variables (even the slot-based model has $O(2^n)$ parameters), and thus their accurate and efficient estimation may not be feasible in practice. It is often assumed implicitly in statistical natural language processing that case slots (or the corresponding random variables) are mutually independent. Although assuming that they are mutually independent would drastically reduce the number of parameters (e.g., under the independence assumption, the number of parameters in a slot-based model becomes $O(n)$), as illustrated above, this assumption is not necessarily valid in practice.

What seems to be true in practice is that some case slots are in fact dependent on one another, but that the overwhelming majority of them are mutually independent, due partly to the fact that usually only a few case slots are obligatory; the others are optional. (Optional case slots are not necessarily independent, but if two optional case slots are randomly selected, it is very likely that they are independent of one another.) Thus the target joint distribution is likely to be approximatable as the product of lower-order component distributions, and thus has, in fact, a reasonably small number of parameters.

In general, any n-dimensional joint distribution can be written as

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_{m_i} \mid X_{m_1}, \ldots, X_{m_{i-1}})$$

for some permutation $(m_1, m_2, \ldots, m_n)$ of $(1, 2, \ldots, n)$, letting $P(X_{m_1} \mid X_{m_0})$ denote $P(X_{m_1})$. A plausible assumption regarding the dependencies between these random variables is that each variable *directly* depends on at most one other variable. This is one of the simplest assumptions that can be made to relax the independence assumption. For example, if the joint distribution $P(X_1, X_2, X_3)$ over 3 random variables X_1, X_2, X_3 can be written (approximated) as $P(X_1) \cdot P(X_2 \mid X_1) \cdot P(X_3 \mid X_2)$, it (approximately) satisfies such an assumption. We call such a distribution a **dependency forest model**. A dependency forest model can be represented by a dependency forest (i.e., a set of dependency trees), whose nodes represent random variables (each labeled with a number of parameters), and whose directed links represent the dependencies that exist between these random variables. A dependency forest model is thus a restricted form of the Bayesian network (note that it is also an extension of the first-order Markov chain model).

Now we turn to the problem of how to select the best dependency forest model from among all possible ones to approximate a target joint distribution based on input data. This problem has already been investigated in the area of machine learning and related fields. In particular, Suzuki (1993) has devised an algorithm, based on the Minimum Description Length (MDL) principle (Rissanen 1989), which estimates the target joint distribution as a dependency forest model (see Li [1998] for a derivation of this algorithm from MDL). Figure 1 shows this algorithm, for which k_i and k_j denote,

```
Algorithm:
Let $T := \emptyset$, $V = \{\{X_i\}, i = 1, 2, \cdots, n\}$;
Calculate the value $\theta(X_i, X_j)$ for all node pairs $(X_i, X_j)$;
$\theta(X_i, X_j) = I(X_i, X_j) - (k_i - 1) \cdot (k_j - 1) \cdot \frac{\log N}{2 \cdot N}$, where mutual information
$I(X_i, X_j) = \sum_{x_i \in X_i, x_j \in X_j} \left( P(x_i, x_j) \log P(x_i, x_j) - P(x_i, x_j) \log(P(x_i) \cdot P(x_j)) \right)$
Sort the node pairs in descending order of $\theta$, and store them into queue $Q$;
while $(\max_{(X_i, X_j) \in Q} \theta(X_i, X_j) > 0)$ do
    Remove $\arg\max_{(X_i, X_j) \in Q} \theta(X_i, X_j)$ from $Q$;
    if $X_i$ and $X_j$ belong to different sets $W_1, W_2$ in $V$
    then Replace $W_1$ and $W_2$ in $V$ with $W_1 \cup W_2$, and add edge $(X_i, X_j)$ to $T$;
Output $T$ as the set of edges of the dependency forest.
```

Figure 1
The learning algorithm.

$N = 9$
$k_{arg1} = 2$
$k_{arg2} = 3$
$k_{from} = 2$
$k_{to} = 2$

θ	X_{arg1}	X_{arg2}	X_{from}	X_{to}
X_{arg1}		-0.28	-0.16	-0.18
X_{arg2}			0.11	0.57
X_{from}				0.28
X_{to}				

$$\begin{tikzcd} & X_{arg2} \arrow[dr] & \\ X_{arg1} & & X_{to} \arrow[dl] \\ & X_{from} & \end{tikzcd}$$

Figure 2
A dependency forest as case frame patterns.

respectively, the number of possible values assumed by random variables X_i and X_j, and N the input data size. We employ Suzuki's algorithm to learn case frame patterns as a dependency forest model.

Let us now consider an example to see how the algorithm works. Suppose that the input data is as given in Table 1 and there are four nodes (random variables) X_{arg1},X_{arg2},X_{from}, and X_{to}. The table in Figure 2 shows the θ values for all node pairs. Our program actually induces the set of possible values of a random variable from those in the input data and thus here $k_{arg1} = 2$. Probabilities are calculated based on maximum-likelihood estimation. Also, the base of logarithm is 2, and $0 \cdot \log 0 = 0$ is assumed. The dependency forest in Figure 2 is then constructed. The dependency forest indicates that there is dependency between the *to* slot and the arg2 slot, and between the *to* slot and the *from* slot, but the arg1 slot is independent from all others. Note that this algorithm in fact outputs a forest consisting of **labeled free trees.** A labeled free tree is a tree in which each node is uniquely associated with a label and the root node is left unspecified. After applying the algorithm, any node can be designated as the root of that tree, since the dependency models based on the same labeled free tree are all equivalent (cf. Li 1998).

3. Experimental Evaluation

3.1 Experiment 1: Slot-based Model

In the first experiment, we used the proposed method to learn slot-based dependencies. As training data, we used the entire bracketed data of the Wall Street Journal corpus (Penn Treebank). We extracted case frame data from the corpus using some heuristic rules. There were 3,678 verbs for which case frame instances were extracted. We considered only the 354 verbs for which more than 50 case frame instances were extracted, since dependencies are not likely to be found with smaller data sizes (see Section 3.4.) Also, we only considered the 12 most frequently occurring case slots (arg1, arg2, *on, in, for, at, by, from, to, as, with, against*) and ignored the others.

Example Case Frame Patterns. We acquired slot-based case frame patterns for the 354 verbs using our method. There were on the average $484/354 = 1.4$ dependency links acquired for each of the 354 verbs. We found that there were some verbs whose arg2 slot is dependent on a preposition (referred to as p) slot. Table 2 shows some of the verbs with large $P(X_{\text{arg2}} = 1, X_p = 1)$ values. We also found that there were some verbs having preposition slots that depend on each other (referred to as $p1$ and $p2$). Table 2 also shows some of the verbs with large $P(X_{p1} = 1, X_{p2} = 1)$ values. The dependencies found by our method seem to agree with human intuition.

Table 2
Verbs and their dependent case slots.

Verb	Dependent Slots	Example
gain	arg2 to	gain 10 to 100
compare	arg2 with	compare profit with estimate
invest	arg2 in	invest share in fund
convert	arg2 to	convert share to cash
add	arg2 to	add 1 to 3
withdraw	arg2 from	withdraw application from office
prepare	arg2 for	prepare case for trial
file	arg2 against	file suit against company
sell	arg2 to	sell facility to firm
lose	arg2 to	lose million to 10%
range	from to	range from 100 to 200
climb	from to	climb from million to million
rise	from to	rise from billion to billion
shift	from to	shift from stock to bond
soar	from to	soar from 10% to 20%
fall	from to	fall from million to million
increase	from to	increase from million to million
climb	from in	climb from million in period
apply	to for	apply to commission for permission
boost	from to	boost from 1% to 2%

Perplexity Reduction. We evaluated the acquired case frame patterns (using the slot-based models) for all of the 354 verbs in terms of reduction in test data perplexity.[2] We conducted the evaluation through a 10-fold cross validation. That is, to acquire case frame patterns for a given verb, we used nine-tenths of the case frames for that verb as training data, saving what remained for use as test data, and then calculated the test data perplexity. We repeated this process 10 times and calculated the average perplexity. We then compared this with the average perplexity for independent models, which were acquired based on the assumption that each case slot is independent.

The experimental results indicate that for some verbs the use of dependency forest models results in a reduction of perplexity as compared to that of independent models. For 30 of the 354 verbs (8%), perplexity reduction exceeded 10%, while the average perplexity reduction overall was only 1%. Nonetheless, it seems safe to say that, with the currently available amount of data, the dependency forest model is more suitable as a representation for the *true* model of case frames than the independent model, at least for 8% of the 354 verbs.

3.2 Experiment 2: Slot-based Disambiguation

In order to quantitatively evaluate the acquired knowledge of case slot dependencies, we conducted a PP-attachment disambiguation experiment, in which we make a decision of the following type: Whether *from Tokyo* should be attached to *fly* or *jet* in the sentence *he will fly a jet from Tokyo.*

2 Test data perplexity is a measure of how well an estimated probability model predicts future data, and is defined as $2^{H(P_T,P_M)}, H(P_T,P_M) = -\sum_x P_T(x) \cdot \log P_M(x)$, where $P_M(x)$ denotes the estimated model, $P_T(x)$ the empirical distribution of the test data. It is in general the case that the smaller perplexity a model has, the closer to the true model it is.

A simple way to disambiguate is to compare the following likelihood values, based on acquired slot-based models,

$$P_{\text{fly}}(X_{\text{arg2}} = 1, X_{\text{from}} = 1) \cdot P_{\text{jet}}(X_{\text{from}} = 0),$$

and

$$P_{\text{fly}}(X_{\text{arg2}} = 1, X_{\text{from}} = 0) \cdot P_{\text{jet}}(X_{\text{from}} = 1)$$

assuming that there are only two case slots arg2 and *from* for the verb *fly*, and there is only one case slot *from* for the noun *jet*. This means that we need to compare

$$\begin{aligned} & P_{\text{fly}}(X_{\text{from}} = 1 \mid X_{\text{arg2}} = 1) \cdot \left(1 - P_{\text{jet}}(X_{\text{from}} = 1)\right) \\ & = P_{\text{fly}}(X_{\text{from}} = 1 \mid X_{\text{arg2}} = 1) - P_{\text{fly}}(X_{\text{from}} = 1 \mid X_{\text{arg2}} = 1) \cdot P_{\text{jet}}(X_{\text{from}} = 1) \end{aligned}$$

and

$$\begin{aligned} & \left(1 - P_{\text{fly}}(X_{\text{from}} = 1 \mid X_{\text{arg2}} = 1)\right) \cdot P_{\text{jet}}(X_{\text{from}} = 1) \\ & = P_{\text{jet}}(X_{\text{from}} = 1) - P_{\text{fly}}(X_{\text{from}} = 1 \mid X_{\text{arg2}} = 1)) \cdot P_{\text{jet}}(X_{\text{from}} = 1). \end{aligned}$$

Since the second term is common to both expressions, we actually only need to compare

$$P_{\text{fly}}(X_{\text{from}} = 1 \mid X_{\text{arg2}} = 1)$$

and

$$P_{\text{jet}}(X_{\text{from}} = 1).$$

Obviously, if we assume that case slots are independent, then we only need to compare $P_{\text{fly}}(X_{\text{from}} = 1)$ and $P_{\text{jet}}(X_{\text{from}} = 1)$. This is nearly equivalent to the disambiguation method proposed by Hindle and Rooth (1991). Their method, referred to here as the Lexical Association (LA) method, actually compares the two probabilities by means of hypothesis testing. Specifically, it calculates the so-called t-score, which is a statistic about the difference between the two probabilities.

Here, we first employ the proposed dependency learning method to judge if slots X_{arg2} and X_{from} with respect to verb *fly* are mutually dependent. Then, if they are dependent, we make the disambiguation decision based on the t-score between $P_{\text{fly}}(X_{\text{from}} = 1 \mid X_{\text{from}} = 1)$ and $P_{\text{jet}}(X_{\text{from}} = 1)$. Otherwise, we consider the two slots independent and make a decision based on the t-score between $P_{\text{fly}}(X_{\text{from}} = 1)$ and $P_{\text{jet}}(X_{\text{from}} = 1)$. We call this disambiguation method DepenLA, since it is a natural extension of LA.

First, we randomly selected the files under one directory of the Wall Street Journal corpus, containing roughly 1/26 of the entire bracketed corpus data, and extracted (v, n_1, p, n_2) quadruples (e.g., (fly,jet,from,Tokyo)) as test data. We then extracted case frames from the remaining bracketed corpus data as we did in Experiment 1 and used them as training data. We repeated this process 10 times and obtained 10 data sets consisting of different training data and test data. In each training data set, there were roughly 128,000 case frames on the average for verbs and roughly 59,000 case frames for nouns. On the average, there were 820 quadruples in each test data set.

We used these 10 data sets to conduct cross-validation on the disambiguation accuracy. We used the training data to acquire dependency forest models, which we then used to perform disambiguation on the test data based on DepenLA. We also tested the performance of LA for comparison. We set the threshold for the t-score to 1.28 for both methods. In both cases, some quadruples remained whose attachment

Table 3
PP-attachment disambiguation results.

Method	Accuracy(%)	Accuracy for 11% of Data (%)
Default	56.2	55.3
LA	78.1 ± 0.5	93.8 ± 0.8
DepenLA	78.4 ± 0.5	97.5 ± 0.5

sites could not be determined. In such cases, we made a default decision—namely, we forcibly attached (p, n_2) to v. This is because we found empirically for our data set that for what remains after applying LA or DepenLA, it is most likely that (p, n_2) goes with v. Table 3 summarizes the results, which are evaluated in terms of disambiguation accuracy, averaged over the ten trials. Table 3 also gives the standard deviation ($\pm$) of each accuracy value, calculated based on the assumption that the 10 cross-validation trials are statistically independent.

We found that as a whole it is still difficult to judge if DepenLA significantly improves upon LA, indicating that it is almost justifiable to rely on the independence assumption in practice, at least when the data size is at the level that is currently available. For about 11% of the verbs, however, for which the dependencies are detected and are strong enough (i.e., $P(X_p = 1 \mid X_{\text{arg2}} = 1) > 0.2$ or $P(X_p = 1 \mid X_{\text{arg2}} = 1) < 0.002$), DepenLA significantly improves upon LA. (The cutoff points were set heuristically by observing the obtained dependencies, but not after the accuracy was calculated.) It is interesting to note that on the subset of data for which DepenLA is found to significantly improve upon LA, LA is already doing quite well: 93.8% as compared to 78.1%. We found that in cases in which dependencies are detected (i.e., $P(X_p = 1 \mid X_{\text{arg2}} = 1) \gg P(X_p = 1)$ or the converse), the probability value $P(X_p = 1)$ is already highly discriminative, that is, either very large or very small. This is probably due to the fact that the verbs for which dependencies are detected are those for which the amount of training data is sufficient (relative to the inherent difficulty of disambiguation for that verb), and hence that are easy to disambiguate.

3.3 Experiment 3: Class-based Model

We also used the proposed method to acquire case frame patterns as class-based dependency forest models, using the 354 verbs in Experiment 1. As before, we considered only the 12 most frequent slots. We generalized the values of the case slots within these case frames using the method proposed in Li and Abe (1998) to obtain class-based case frame data. We then used these data as input to the learning algorithm. The results were rather discouraging: very few case slots were determined to be dependent in the case frame patterns. To be more precise, there were on the average only $64/354 = 0.2$ dependency links found for each verb. This is because the number of parameters in a class-based model was large as compared to the size of the data we had available.

These experimental results seem to justify the commonly made assumption that class-based case slots, and hence word-based case slots, are mutually independent, when the data size available is at the level of what is currently provided by the Penn Treebank.

3.4 Experiment 4: Simulation

In order to test how large a data size is required to estimate a dependency forest model, we conducted the following experiment. We defined an artificial model in the form of

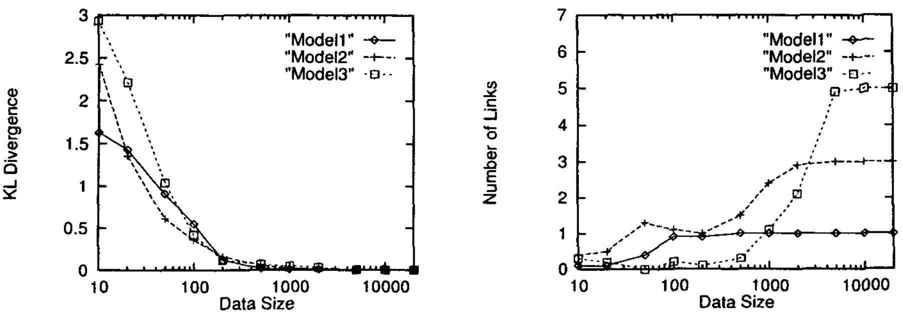

Figure 3
Simulation results.

a dependency forest model and generated data according to its distribution. We then used the data obtained to estimate a model, and evaluated the estimated model by measuring the KL divergence between the estimated model and the true model. We also checked the number of dependency links in the obtained model. We repeatedly generated data and observed the learning curve, namely the relationship between the data size and the KL divergence between the estimated and true models, and the relationship between the data size and the number of dependencies in the estimated model. We repeated this experiment on three artificial models. Figure 3 shows the results of these experiments averaged over 10 trials. The number of parameters in Model 1, Model 2, and Model 3 are 18, 30, and 44 respectively, and the number of links in them 1, 3, and 5. Note that the KL divergence between the estimated model and the true model converges to 0, as expected. Also note that the number of links in the estimated model converges to the correct value (1, 3, and 5) in each of the three examples.

These simulation results verify that the dependencies between case slots can be accurately learned when there is enough data, given that the true model is representable as a dependency forest model. We also see that to estimate a model reasonably accurately, the data size required is as large as 5 to 10 times the number of parameters. For example, for the KL divergence to go below 0.1, we need more than 200 examples, which is roughly 5 to 10 times the number of parameters. (Recall that there were only 354 verbs, having frequencies greater than 50 in our experiments, see Subsection 3.1.)

In Experiment 2, there were 12 binary-valued random variables associated with each verb, and hence its distribution is thought to be approximatable by a model with a comparable number of parameters. So having 50 to 100 examples for each of these verbs approaches the minimum data size required for reasonable estimation of their dependencies. In Experiment 3 there were also 12 slots, but each slot could take on roughly 10 classes as its values and thus a class-based model tended to have about 120 parameters. The corpus data available to us was in no way sufficient for this case.

4. Summary

We have proposed a method of learning dependencies between case slots, based on an estimation method for dependency forest models. When using slot-based models, it was found empirically that some case slots are dependent, and when such dependencies are detected, using that knowledge of dependencies can significantly improve PP-attachment disambiguation accuracy. For class-based models, however, most case slots were judged independent. These empirical findings indicate that the assumption often made in practice that case slots in a class-based model are mutually independent is indeed justified, at least with the data size currently available in the Penn Treebank.

Acknowledgments
We are grateful to Mr. K. Nakamura, Mr. T. Fujita, and Dr. S. Doi of NEC and Ms. Y. Yamaguchi of NIS for their encouragement and collaboration. We thank the anonymous CL reviewers for their valuable comments and criticisms.

References

Collins, Michael and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. *Proceedings of the 3rd Workshop on Very Large Corpora.*

Hindle, Donald and Mats Rooth. 1991. Structural ambiguity and lexical relations. *Proceedings of the 29th Annual Meeting,* pages 229–236. Association for Computational Linguistics.

Li, Hang. 1998. *A Probabilistic Approach to Lexical Semantic Knowledge Acquisition and Structural Disambiguation.* Ph.D. Thesis, University of Tokyo.

Li, Hang and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics,* 24(2):217–244.

Rissanen, Jorma. 1989. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Co., Singapore.

Suzuki, Joe. 1993. A construction of Bayesian networks from databases based on an MDL principle. *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence,* pages 266–273.