Spring 2010 LTI Report

Probabilistic Ontology Model

Ni Lao

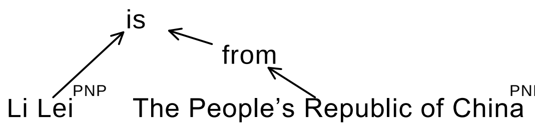
1. Introduction

This is a critique for Tom M. Mitchell's talk about the Read the Web project. In the talk, Tom described the research to develop a never-ending language learner that runs 24 hours per day, forever. The system is able to extract more information from the web to populate its growing knowledge base of structured knowledge. One limitation to the system, in my opinion, is the need for human to supply ontology. This is less applicable if we want to apply the method to many new domains, or to very large domains. Therefore, it is very important to develop methods that aid the process of constructing ontology for a new domain. Here I am giving a sketchy design of a system which can automatically extract ontology from parsed text. I will first define a probabilistic ontology model based on paths on the ontology graph, and then I will describe efficient inference, parameter estimation, and structure induction procedures for this model. Finally, I will give an experiment design on Penn TreeBank data.

2. Probabilistic Ontology Model (POM)

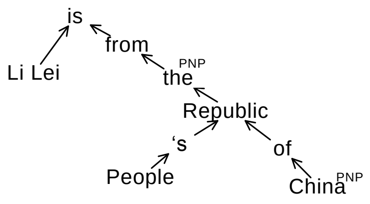
2.1. Text Representation with Dependency Trees

We assume that text is represented as a set of dependency parse trees, with proper nouns and common nouns annotated. For example the sentence "Li Lei is from People's Republic of China" is annotated as



Following previous works on named entity extraction [] and semantic role labeling [], a pattern on the parse tree can be defined as either the path between any two nouns (e.g. XXX →is ←from ←XXX), or the path from a noun to any of its ancestors (e.g. is ←from ←XXX, from ←XXX).

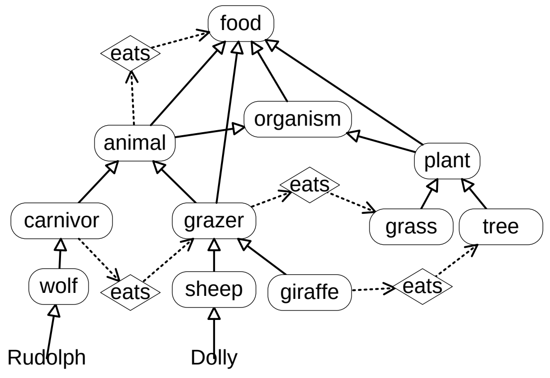
Alternatively, we may also consider a more complex representation, which break compound nouns into single words. This representation may help leverage the relationships between words in a compound noun: for example "horse leg" indicates that leg is a part of horse.



2.2. Representing Ontology

An ontology **O**={**C**, **E**, **A**} is a tuple of a set of concepts **C**={c_i}, a set of entities **E**={e_j}, and a set of attributes **A**={a_k}. We assume that each concept *c* approximately corresponds to a common noun phrase (CNP) (e.g. mouse, mountain), each entity *e* approximately corresponds to a proper noun phrase (PNP) (e.g. Mickey, Everest), and each attribute *a* approximately corresponds to a possible relation between concepts (e.g. CEOof(e_1, e_2) representing that e_1 is the CEO of e_2). Each concept is associated with a set of attributes c.**A**.

We assume that the concepts form hierarchical (acyclic) structures, with each node inheriting all the attributes of its ancestors. This quality will be more concrete after we describe the random walk based probabilistic model.



2.3. A Probabilistic Model Based on Random Walk on Ontology Graph

The problems of eliciting an ontology **O** with its parameter θ can be seem as maximizing the log likelihood of a corpus D given this ontology. Given a corpus D={$s^{(m)}$}, where each sentence s_i is a parse tree annotated with CNP and PNP tags, we define our objective function as the regularized probability over the training data

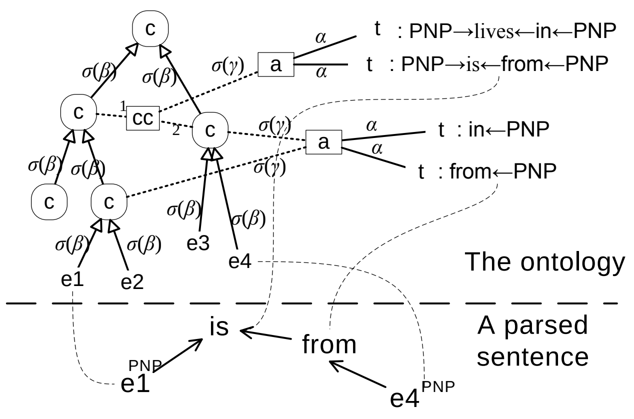
$$L(\boldsymbol{\theta}) = \sum_{m=1..M} \log P(s_m; \theta) - \lambda_1 |\boldsymbol{\theta}|_1 - \lambda_2 |\boldsymbol{\theta}|_2 / 2 \quad (1)$$

The probability of each sentence is defined as the probability of its entities.

$$P(s^{(m)}; \theta) = \frac{\exp(\theta^T \mathbf{f}(\mathbf{e}^{(m)}, t^{(m)}))}{\sum_{\mathbf{e}} \exp(\theta^T \mathbf{f}(\mathbf{e}, t^{(m)}))} \quad (2)$$

where $\mathbf{e}^{(m)} = (e_1^{(m)}, ..., e_n^{(m)})$ is a vector of entities in the m-th sentence. Here we treat each entity as a categorical random variable which has domain as all the noun phrases (PNPs and CNPs) in the corpus.

The whole learning process is very much like the problem of co-clustering the set of all entities (pairs) and patterns in the sentences. We introduce three sets of parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$, where $\boldsymbol{\alpha}$ correspond to the association from patterns to attributes, $\boldsymbol{\beta}$ correspond to the associations from entities (pairs) to concepts (pairs), and $\boldsymbol{\gamma}$ correspond to the association between concept (pairs) and attributes.



We define a *composite concept* as the ordered tuple of any two basic concepts in the ontology. A *unary path* $p \in \mathbf{P}(t,e)$ is defined as one of the paths starting from pattern t and end at entity e, where t is a unary pattern. A corresponding *unary feature* f_p fires only if both $p.e$ and $p.t$ are matched on the parse tree of a sentence. Its weight is defined as θ_f, the multiplication of all the ontology parameters (α's, β's, and γ's) along the path.

Similarly, a *pairwise path* $p \in \mathbf{P}(t,e1,e2)$ is defined as one of the hyper-paths which start from a pairwise pattern t and end at entities $e1$ and $e2$. And its corresponding *pairwise feature* f_p has weight θ_p, the multiplication of all the ontology parameters (α's, β's, and γ's) along the path.

$$\theta_p = \prod_{\alpha \in \boldsymbol{\alpha}} \alpha^{c_{p,\alpha}} \prod_{\beta \in \boldsymbol{\beta}} \sigma(\beta)^{c_{p,\beta}} \prod_{\gamma \in \boldsymbol{\gamma}} \sigma(\gamma)^{c_{p,\gamma}} \tag{3}$$

where $a_{f,\alpha}$ is the number of times α appears in feature f. $b_{f,\beta}$ and $c_{f,\gamma}$ are defined similarly. We use the stretched hyperbolic tangent function $\sigma(x)=tanh(x/2)=(e^x-1)/(e^x+1)$ to transform the range of β's and γ's from R^+ to [0,1). In this way, it is easier to interpret their semantics: whether a class c inherits another class c', or whether a class c has the attribute a. It is also preferable in the sense that the sparsity of parameters are maintained: $\sigma(x)=0$ if $x=0$.

To simplify our notations, we define $w_\alpha = \alpha$, ' $w_\beta = \sigma(\beta)$, $w_\gamma = \sigma(\gamma)$. Then Eq (3) can be simplified as

$$\theta_p = \prod_{w \in \mathbf{w}} w^{c_{p,w}} = \prod_{w \in p} w \tag{4}$$

3. Learning POM

3.1. Inference

To simply inference we use Pseudo Log Likelihood (PLL) to approximate the joint probability defined by Eq. (2). Basically, we predict one NP in the sentence conditioned on the rest of the sentence. One can show that as number of training data goes to infinity, the parameters estimated from Maximum Conditional Likelihood Estimation (MLE) and maximum Conditional Pseudo-Likelihood Estimation (MCPE) become the same[]. The probability of observing the m-th sentence $s^{(m)}$ is defined as

$$P(s^{(m)};\theta)=\prod_i P(e_i^{(m)}\mid s^{(m)}/e_i^{(m)};\theta)=\prod_i \frac{1}{Z_i^{(m)}(\theta)}\exp(\theta^T\mathbf{f}(e_i^{(m)},s^{(m)}/e_i^{(m)}))$$
$$Z_i^{(m)}(\theta)=\sum_{e_i}\exp(\theta^T\mathbf{f}(e_i,s^{(m)}/e_i^{(m)}))=N-|\mathbf{E}_i^{active}|+\sum_{e_i\in\mathbf{E}_i^{active}}\exp(\theta^T\mathbf{f}(e_i,s^{(m)}/e_i^{(m)})) \quad (5)$$

, where $e_i^{(m)}$ denote its i-th entity, N is the total number of entities in the ontology, and $\mathbf{E}_i^{\text{active}}$ is the set of entities which, if placed in the i-th slot, can activate any of the features. We use search process to find $\mathbf{E}_i^{\text{active}}$ and the set of activated features. We define the per-instance objective function as

$$l_i^{(m)}(\theta)=\log P(e_i^{(m)}\mid s^{(m)}/e_i^{(m)};\theta) \quad (6)$$

Algorithm 1 illustrates the procedure to find $\mathbf{E}_i^{\text{active}}$. We first consider any unary pattern t matched in the sentence with $e_i^{(m)}$ as it's argument. We can find all paths corresponding to its down stream entities in the ontology by depth first search. Then we consider any pairwise pattern t matched in the sentence with $e_i^{(m)}$ as one of its argument and $e_j^{(m)}$ as another argument.

```
Algorithm 1: Finding Activated Paths/Entities
Input: sentence s
Output: the set of activated entities {$\mathbf{E}_i^{\text{active}}$}
Initialize: {$\mathbf{E}_i^{\text{active}} = \varnothing$}
for any unary pattern t matched in s at the i-th slot
    for any its immediately related concept c
        $\mathbf{E}_i^{\text{active}} = \mathbf{E}_i^{\text{active}} \cup$ Decendent(c)
for any pairwise pattern t matched in s at the i-th and j-th slot
    for any its immediately related composite concept cc
        if cc.child1 $\in$ Ancestor($e_i^{(m)}$)
            $\mathbf{E}_j^{\text{active}} = \mathbf{E}_j^{\text{active}} \cup$ Decendent(cc.child2)
        if cc.child2 $\in$ Ancestor($e_j^{(m)}$)
            $\mathbf{E}_i^{\text{active}} = \mathbf{E}_i^{\text{active}} \cup$ Decendent(cc.child1)
return {$\mathbf{E}_i^{\text{active}}$}
```

For a large ontology, the number of paths (features) can be very large. Explicitly enumerate each of them can be very inefficient. Here we develop a dynamic programming procedure to estimate the gradient of w's. Let's define several useful notations first. Define $\theta_{f|w\rangle}$ as the product of parameters before w (upstream, close to the patterns) on path f, and $\theta_{f|\langle w}$ the product of parameters after w (downstream, close to the entities) on path f. Then, we can decompose $\theta_f / w = \theta_{f|\langle w}\theta_{f|w\rangle}$. Define

$$\begin{aligned}
\theta_{a\langle w} &= \sum_{f.b=w,\ f.e=a} \theta_f \\
\theta_{\langle w} &= \sum_{f.b=w,f.e\in\mathbf{E}} p(f.e;\theta)\theta_f \\
\theta_{w\rangle\mathbf{P}} &= \sum_{f.e=w,f.b\in\mathbf{P}} \theta_{f|w\rangle} \\
\theta_{w\rangle\mathbf{P},b} &= \sum_{w\in f,f.p\in\mathbf{P},b\in f|w\rangle} \theta_{f|w\rangle}
\end{aligned} \quad (7)$$

These quantities have the following relations, which only depend on localized information, and therefore enable dynamic programming strategy:

$$\begin{aligned}
\theta_{a\langle w} &= \sum_{e\in w.\mathbf{E}} I(e=a) + \sum_{w'\in down(w)} w'\theta_{a\langle w'} \\
\theta_{\langle w} &= \sum_{e\in w.\mathbf{E}} p(e;\theta) + \sum_{w'\in down(w)} w'\theta_{\langle w'} \\
\theta_{w\rangle \mathbf{P}} &= \sum_{w'\in up(w)} w'\theta_{w'\rangle \mathbf{P}} \\
\theta_{w\rangle \mathbf{P},b} &= \sum_{w'\in up(w)} w'\theta_{w'\rangle \mathbf{P}} + \sum_{cc\in w.\mathbf{CC}} \theta_{b\langle cc\backslash w}\theta_{cc\rangle \mathbf{P}}
\end{aligned} \tag{8}$$

where *w.e₁* and *w.e₂* are the entity on the downstream and upstream side of link *w* respectively, and *w.***CC** is the set of composite concepts that use *w.e₂* as one of its argument.

3.2. Parameter Estimation

Generally for any Markov random fields, the gradient of log-likelihood can be written as

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \theta} = \langle \mathbf{f} \rangle_{\tilde{p}(\mathbf{x})} - \langle \mathbf{f} \rangle_{p(\mathbf{x};\boldsymbol{\theta})} \tag{9}$$

where $\langle \cdot \rangle_p$ is the expectation under distribution *p*, and $\tilde{p}$ is the empirical distribution. Therefore,

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \theta_f} = \begin{cases} I(f.e = e_{tgt}) - p(f.e;\theta) & f \in \mathbf{F}_{active} \\ 0 & else \end{cases} \tag{10}$$

Here we eclipse the super script *m* **and subscript *i* **for clarity.
The gradients of α's, β's, and γ's can be estimated by the chain rule

$$\begin{aligned}
\frac{\partial l(\boldsymbol{\theta})}{\partial w} &= \sum_{f\in \mathbf{F}^{active}} \left[I(f.e = e_{tgt}) - p(f.e;\theta) \right] \frac{\theta_f}{w} \\
\frac{\partial l(\boldsymbol{\theta})}{\partial \alpha} &= \frac{\partial l(\boldsymbol{\theta})}{\partial w_\alpha} \\
\frac{\partial l(\boldsymbol{\theta})}{\partial \beta} &= \frac{\partial l(\boldsymbol{\theta})}{\partial w_\beta} \frac{2e^\beta}{(e^\beta+1)^2} \\
\frac{\partial l(\boldsymbol{\theta})}{\partial \gamma} &= \frac{\partial l(\boldsymbol{\theta})}{\partial w_\gamma} \frac{2e^\gamma}{(e^\gamma+1)^2}
\end{aligned} \tag{11}$$

Now we can rewrite the gradient in Eq.(8) as:

$$\begin{aligned}
\frac{\partial l_i(\boldsymbol{\theta})}{\partial w} &= g(e_i) + \sum_{j\neq i} g(e_i, e_j) \\
g(a) &= \theta_{a\langle w}\theta_{w\rangle \mathbf{P}(a)} - \theta_{\langle w}\theta_{w\rangle \mathbf{P}(a)} \\
g(a,b) &= \theta_{a\langle w}\theta_{w\rangle \mathbf{P}(a,b),b} - \theta_{\langle w}\theta_{w\rangle \mathbf{P}(a,b),b}
\end{aligned} \tag{12}$$

We can use second order optimization procedure like orthant-wise L-BFGS (Andrew & Gao, 2007) to estimate the parameters.

3.3. Structure Learning

The problem of learning the structure of ontology has three sub problems: discover new rules, discover new concepts, and discovery new attributes. The details of them are to be designed.

4. Experiment

Initial experiment can be conducted on small but well controlled corpus like Penn Tree Bank.