HIDDEN VECTOR STATE MODEL FOR HIERARCHICAL SEMANTIC PARSING

Yulan He *Steve Young*

Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ
United Kingdom

ABSTRACT

The paper presents a Hidden Vector State (HVS) model for hierarchical semantic parsing. The model associates each state of a push-down automata with the state of a HMM. State transitions are factored into separate stack pop and push operations and then constrained to give a tractable search space. The result is a model which is complex enough to capture hierarchical structure but which can be trained automatically from unannotated data. Experiments have been conducted on ATIS-3 1993 and 1994 test sets. The results show that the HVS model outperforms a general finite state tagger (FST) by 19% to 32% in error reduction.

1. INTRODUCTION

Semantic parsing involves analysing an utterance in sufficient detail to enable the relevant information to be extracted. In the context of spoken dialogue systems, this function has typically been implemented via hand-crafted semantic grammar rules and some form of robust parser [1]. However, this approach carries a high development cost and it can also lead to fragile operation since users do not typically know what grammatical constructions are supported by the system.

An alternative approach is to use statistical methods to map directly from word strings into the intended meaning structures. In this approach, explicit grammar rules are replaced by a statistical model of sentence generation [2]. The generative power of such models extend from finite-state *regular* languages to recursive-hierarchical *context-free* languages. Examples of the former include the finite state semantic tagger used in AT&T's CHRONUS [3] and of the latter include recursive models such as BBN's Hidden Understanding Model (HUM) [4]. In practice it is found that both these extremes have limitations. Finite state taggers (FSTs) are unable to represent embedded structural information and thus fail to make important distinctions (eg confusing the departure and destination in a travel request). Recursive-hierarchical models can represent such dependencies but they are not easily trained from unlabelled data [5], instead they typically require fully annotated treebank data [6].

This paper describes a class of models, called Hidden Vector State (HVS) models, which extend simple discrete hidden Markov models to enable hierarchical structure to be efficiently represented whilst retaining the computational tractability of regular HMMs. The experiments described below show both that this model can be trained on unannotated data using EM and that it can learn long range dependencies sufficient to resolve semantic ambiguities.

The remainder of this paper is organized as follows: Section 2 describes the HVS model and how it can be trained from unannotated data using EM and the target semantics as a constraint. Section 3 then presents experimental results obtained using the ATIS database. Finally, Section 4 presents our conclusions regarding the HVS model.

2. HIDDEN VECTOR STATE MODEL

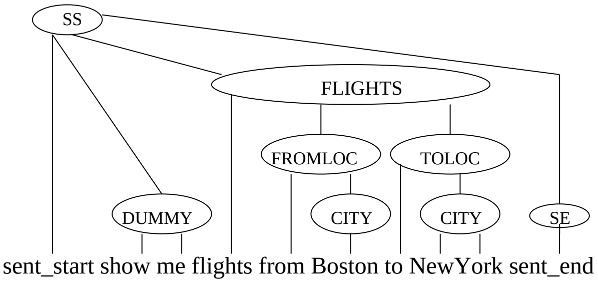

Fig. 1. Example of a parse tree.

The hidden vector state (HVS) model is best introduced by an example. Consider the parse tree shown in Figure 1, the semantic information relating to any single word can be stored as a vector of semantic tag names starting from the preterminal tag and ending at

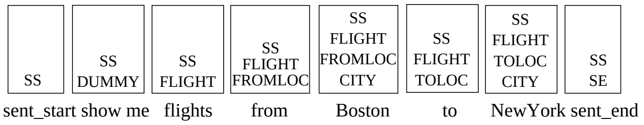

Fig. 2. Vector state equivalent of the parse tree.

the root tag. For example, the word *Boston* is described by the semantic vector *[CITY, FROMLOC, FLIGHT, SS]* and the complete parse tree can be replaced by a sequence of vectors as shown in Figure 2. Viewing each vector state as a hidden variable, the whole parse tree can again be treated as a Markov process, this is the *Hidden Vector State (HVS)* model.

When viewed as a generator, each state in a HVS model represents the state of the stack in a push down automata. If state transitions were unconstrained the result would be a model with equivalent power to a fully recursive hierarchical HMM (ie it would be context free). However, transitions between states can be factored into a stack shift followed by a push of one or more semantic concepts relating to the next input word. Each operation in this two stage factorisation can then be constrained. In particular, if the stack size is limited and the number of new concepts to be pushed for each new word is limited to one, then the resulting parser is effectively a form of discrete Markov model extended to include an efficient history mechanism. Viewed from the perspective of parsing, the model supports embedding but all parse trees are strictly right branching.

More formally, let the complete set of model parameters be denoted by λ and let each state at time t be denoted by a vector of D_t semantic concept labels (tags) $\mathbf{c}_t = [c_{1t}, c_{2t}, ..c_{D_t t}]$ where c_{1t} is the preterminal concept and $c_{D_t t}$ is the root concept (SS in Figure 2). Given a word sequence W, concept vector sequence $\mathbf{C}$ and sequence of stack pop operations N, the joint likelihood function is defined as:

$$L(\lambda) = \log P(W, \mathbf{C}, N|\lambda) \tag{1}$$

EM-based parameter estimation then aims to maximize the expectation of $L(\lambda)$ given the observed data and current estimates. To do this, define the auxiliary Q function as:

$$\begin{aligned} Q(\lambda|\lambda^k) &= \mathrm{E}\ [\log P(W, \mathbf{C}, N|\lambda)|W, \lambda^k] \\ &= \sum_{\mathbf{C},N} P(\mathbf{C}, N|W, \lambda) \log P(W, \mathbf{C}, N|\lambda^k) \end{aligned} \tag{2}$$

As discussed above, $P(W, \mathbf{C}, N)$ can be decomposed as follows

$$\begin{aligned} P(W,\mathbf{C}, N) = \prod_{t=1}^{T} P(n_t|W_1^{t-1}, \mathbf{C}_1^{t-1}) \cdot \\ P(c_t[1]|W_1^{t-1}, \mathbf{C}_1^{t-1}, n_t) \cdot P(w_t|W_1^{t-1}, \mathbf{C}_1^{t}) \end{aligned} \tag{3}$$

where the notation W_1^t denotes words $w_1..w_t$ and similarly for concepts. In the version of the HVS model discussed in this paper, equation 3 is approximated by

$$\begin{aligned} P(n_t|W_1^{t-1}, \mathbf{C}_1^{t-1}) &\approx P(n_t|\mathbf{c}_{t-1}) && (4) \\ P(c_t[1]|W_1^{t-1}, \mathbf{C}_1^{t-1}, n_t) &\approx P(c_t[1]|c_t[2 \cdots D_t]) && (5) \\ P(w_t|W_1^{t-1}, \mathbf{C}_1^{t}) &\approx P(w_t|\mathbf{c}_t) && (6) \end{aligned}$$

Thus, the generative process associated with this constrained version of the HVS model consists of three steps for each word position t: (a) choose a value for n_t; (b) select preterminal concept tag $c_t[1]$; (c) select a word w_t.

Our goal in training the HVS model is to avoid the use of word-level annotations on the grounds that whilst it is reasonable to ask an application designer to provide examples of utterances which would yield each type of semantic schema, it is not reasonable to require utterances with manually transcribed parse trees. Thus, for training we assume the availability of a set of domain specific lexical classes and abstract semantic annotations for each utterance. For example, in a flight information system, it is possible to group all city names like *Boston, New York, Denver* etc into one class *CITY.* Such domain specific classes can usually be extracted directly from the domain database schema. Given these classes, all word sequences are then preprocessed and the detected class members are replaced by their corresponding class names. This also effectively reduces the vocabulary size of the model. An abstract semantics provides dominance relationships between individual semantic tags without considering tag sequences or attempting to identify tag/word pairs. For example, all the following sentences share the same semantics:

1. Show me flights arriving in X at T.
2. List flights arriving around T in X.
3. Which flight reaches X before T.

```
FLIGHT(TOLOC(CITY(X),TIME_RELATIVE(TIME(T))))
```

These semantic annotations serve to partially annotate the training data[1]. Experiments using pure abstract semantics (ie without any lexical features) have also been conducted and the result will be presented in Section 3.

[1] Note that the use of lexical items is justified on the grounds that they can be derived automatically from the domain database.

Parameter estimation is based on EM. Equations 4 to 6 are plugged into equation 2 and maximised. State occupation probabilities are then computed using a conventional forward-backward scheme. The abstract annotations are used to constrain the search space by assigning zero probability to any stack configuration which is incompatible with the target semantics.

3. EXPERIMENTS

This section describes the experimental evaluation of the HVS model using the ATIS-3 NOV93 and DEC94 test sets. The training set consists of 4978 utterances selected from the Class A training data in the ATIS-2 and ATIS-3 corpora. In order to evaluate the performance of the HVS model, a simple finite state semantic tagger (FST) has been implemented as a baseline system. Note however that in contrast to the earlier work using an FST conducted by AT&T [3], where maximum likelihood estimation based on relative frequency counts on a fully annotated ATIS training set was used, our FST model only uses abstract annotations for training data, similar to those described for the HVS model.

3.1. Experimental Setup

A set of 30 domain-specific lexical classes were extracted from the ATIS-3 database. The training data was then preprocessed with the detected class members being replaced by their corresponding class names. Abstract semantics for each training utterance were derived semi-automatically from the SQL queries provided in ATIS-3.

In order to evaluate the performance of the model, a frame structure for every test set utterance was also derived consisting of a goal and a number of slot/value pairs. An example of such a reference frame is:

```
Show me flights from Boston to New York.
Goal: FLIGHT
Slots: FROMLOC.CITY = Boston
       TOLOC.CITY = New York
```

Performance was then measured in terms of goal detection accuracy as well as slot/value pair comparison. In the experiments reported here, Bayesian Belief Networks (BN) similar to those proposed in [7] were used to identify each of the 19 distinct goals using as input 47 distinct semantic concepts.

Various smoothing techniques have been tested and it was found that linear discounting for transition probabilities and Good-Turing for output probabilities yield the best result for the FST model, whereas nonlinear discounting for vector state stack operation probabilities and Witten-Bell for output probabilities achieve the highest F-measure score for the HVS model [8, 9].

3.2. Results

The trained FST and HVS models were tested on the NOV93 and DEC94 ATIS-3 test sets. In each case, the parsed results were compared with the slots from the reference frames and the precision (P) and recall (R) values were computed. These two values are then combined to form a single F-measure:

$$F = 2PR/(P + R) \tag{7}$$

Table 1 gives the F-measure scores as well as goal detection accuracy for both the FST and the HVS models. It can be observed that the HVS model outperforms the FST model in both F-measure and goal detection for NOV93 test set and F-measure for DEC94 test set. The goal detection accuracy for DEC94 is about the same for both models. The HVS model has a higher precision and slightly higher recall than those of the FST model, which results in F-measure being increased by 3.61% and 5.37% for NOV93 and DEC94 test sets respectively. The lower precision compared to recall exhibited by the HVS model is a result of incorrect taggings. For example, the tag *FROMLOC.CITY* may be assigned to a word which is not one of the city names. This is a penalty incurred from eschewing the use of fully annotated training data. If the system is trained on pure abstract semantic annotated data (without any lexical features), F-measure for NOV93 and DEC94 are 77.80% and 84.18% respectively, and the goal detection accuracy is 85.91% and 86.52%.

	1993 Test Set		*1994 Test Set*	
Measurement	*FST*	*HVS*	*FST*	*HVS*
Recall	91.23%	91.53%	92.51%	94.69%
Precision	78.27%	83.45%	79.94%	86.44%
F-measure	84.26%	87.30%	85.77%	90.38%
Goal	83.48%	89.29%	88.09%	87.64%

Table 1. Performance comparison of FST and HVS.

The stack depth in the HVS model determines the number of previous semantic tags which can be recorded as history context. Figure 3 shows the effect of varying the maximum stack depth where solid lines represent F-measure and dash lines represent goal detection accuracy [2]. It can be observed that the highest F-measure score and the best goal detection accuracy are obtained when the stack depth is set to 3 for both NOV93 and DEC94 test sets. These two values remain constant for the stack depth 4 and above. The optimal stack depth

[2] As the tag *SS* for sentence start marker always exists in all vector states, it is omitted from vector state stacks for clarity, ie, the actual number of semantic tags kept in a vector state is $n + 1$ for the stack depth n.

is in fact determined by the deepest hierarchical level within the semantic tags that appear in the abstract semantic annotations. Another conclusion we can draw from Figure 3 is that the system performance is similar to that of the FST model when the stack depth is set to 2. The HVS model with stack depth 2 thus corresponds to an FST model with bigram transition probabilities.

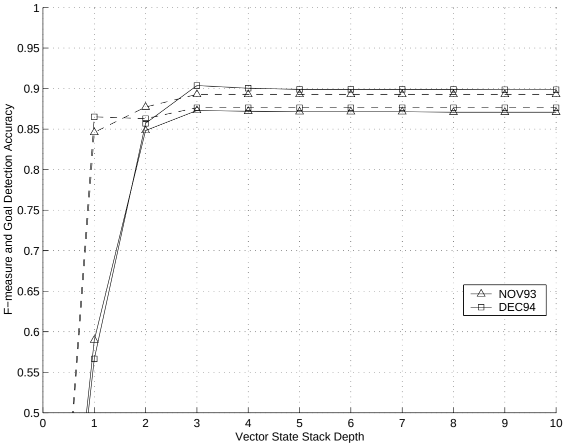

Fig. 3. F-measure and goal detection accuracy vs stack depth.

Another interesting issue is the number of parameters to be estimated in each model. This is related to the total number of distinct states and the vocabulary size. Table 2 gives the relevant statistics for the FST model and the HVS model. Although the state space for the HVS model is large relative to the FST model, the possible transitions between any two states are constrained by the maximum depth of the vector state stack and the condition that only one new semantic tag can be added per input word.

	FST Model	*HVS Model*
Number of States	120	2799
Vocabulary Size	611	611
Total Parameters	87720	1724184

Table 2. Statistics of states and vocabulary size.

4. CONCLUSION

This paper has presented the hidden vector state (HVS) model for hierarchical semantic parsing and its experimental evaluation on the ATIS-3 database. The key features of the model are its ability for representing hierarchical information in a constrained way and its capability for training directly from the target semantics without explicit word-level annotation. The latter is thought to be particularly important since it simplifies the development process and facilitates on-line learning.

Also, unlike more general hierarchical models, the HVS model is well-suited to left-right decoding since all partial paths covering W_1^t can be compared directly without normalisation and pruning since they contain exactly the same number of probabilities. Moreover, the effective state space and model size are much reduced compared to fully recursive hierarchical models.

The experimental results show improved performance of the HVS model relative to a conventional finite state semantic tagger and suggest that a maximum stack depth of 3 is sufficient to capture all necessary hierarchical structure in the ATIS data. Larger and more complex tasks may require deeper stack depths and correspondingly larger search spaces, and we are currently investigating this.

5. REFERENCES

[1] W. Ward and S. Issar, "Recent improvements in the CMU spoken language understanding system," in *Proc. of the ARPA Human Language Technology Workshop.* 1996, pp. 213–216, Morgan Kaufman Publishers, Inc.

[2] S.J. Young, "Talking to machines (statistically speaking)," in *Proc. of Spoken Language Processing*, Denver, Colorado, Sep 2002.

[3] E. Levin and R. Pieraccini, "CHRONUS, the next generation," in *Proc. of the DARPA Speech and Natural Language Workshop*, Austin, TX, Jan. 1995, pp. 269–271, Morgan Kaufman Publishers, Inc.

[4] R. Schwartz, S. Miller, D. Stallard, and J. Makhoul, "Hidden understanding models for statistical setence understanding," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Munich, 1997, pp. 1479–1482.

[5] K. Lari and S.J. Young, "The estimation of stochastic context-free grammars using the inside-outside algorithm," *Computer Speech and Language*, vol. 4, no. 1, pp. 35–56, 1990.

[6] C. Chelba and M. Mahajan, "Information extraction using the structured language model," in *Proc. of Conference on Empirical Methods in Natural Language Processing*, 2001.

[7] H. Meng, W. Lam, and C. Wai, "To believe is to understand," in *Proceedings of the 6th European Conference on Speech Communication and Technology*, 1999.

[8] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependencies in stochastic language modelling," *Computer Speech and Language*, vol. 8, no. 1, pp. 1–38, 1994.

[9] I.H. Witten and T.C. Bell, "The zero frequency problem: estimating the probabilities of novel events in adaptive text compression," *IEEE Trans. on Information Theory*, pp. 1085–1093, 1991.