Use of Negative Examples in Training the HVS Semantic Model

Filip Jurčíček[1], Jan Švec[2], Jiří Zahradil[2], and Libor Jelínek[2]

[1] Center of Applied Cybernetics, University of West Bohemia
Pilsen, 306 14, Czech Republic
filip@kky.zcu.cz
[2] Department of Cybernetics, University of West Bohemia
Pilsen, 306 14, Czech Republic
honzas@kky.zcu.cz, jzahrad@kky.zcu.cz, jelinekl@kky.zcu.cz

Abstract. This paper describes use of negative examples in training the HVS semantic model. We present a novel initialization of the lexical model using negative examples extracted automatically from a semantic corpus as well as description of an algorithm for extraction these examples. We evaluated the use of negative examples on a closed domain human-human train timetable dialogue corpus. We significantly improved the standard PARSEVAL scores of the baseline system. The labeled F-measure (LF) was increased from 45.4% to 49.1%.

1 Introduction

A corpus for semantic parsing usually consists of utterances (word sequences) and its semantic annotation (semantic parse trees). In such corpus, there are positive and negative examples which can be used for training statistical models.

A positive example is a pair of a word and its semantic annotation. A positive example says that some word has some (concrete) semantic annotation. A negative example, similarly to a positive example, is a pair of a word and its semantic annotation; however, it says about a word that it does not have a semantic annotation. A negative example gives us much less information than a positive example because we have to collect several negative examples to replace one positive example.

In this paper, the statistical semantic parsing is a search of the sequence of concepts $S = c_1, c_2, \ldots, c_T$ that has the maximum aposteriori probability $P(S|W)$ for the word observation $W = w_1, w_2, \ldots, w_T$. The search can be described as

$$\begin{aligned} S^* &= \underset{S}{\operatorname{argmax}}\, P(S|W) \\ &= \underset{S}{\operatorname{argmax}}\, P(W|S)P(S) \end{aligned} \tag{1}$$

where $P(S)$ is the semantic model and $P(W|S)$ is the lexical model.

In Section 2, we describe the HVS model with the baseline initialization of a lexical model. Section 3 details both positive and negative examples and the way how to collect negative examples and a new initialization of the lexical model. In Section 4, we provide experimental results. Finally, Section 5 closes this paper.

Petr Sojka, Ivan Kopeček and Karel Pala (Eds.): TSD 2006, LNAI 4188, pp. 605–612, 2006.

2 The HVS Model

The hidden vector state (HVS) model is an approximation of a pushdown automaton. A *vector* state in the HVS model represents a stack of a pushdown automaton, which keeps information that spans over several words.

The semantic information matching every word in an utterance is described by a sequence of concepts from a leaf to a root of a semantic annotation (see Fig. 1). If we place concepts along the way from the leaf to the root to a vector, than a derivation tree can be transformed to a sequence of these vectors. We imposed a hard limit on the maximum depth of a stack equal to four. For example, the word *Prague* is described by the vector state [STATION, TO, DEPARTURE, EMPTY].

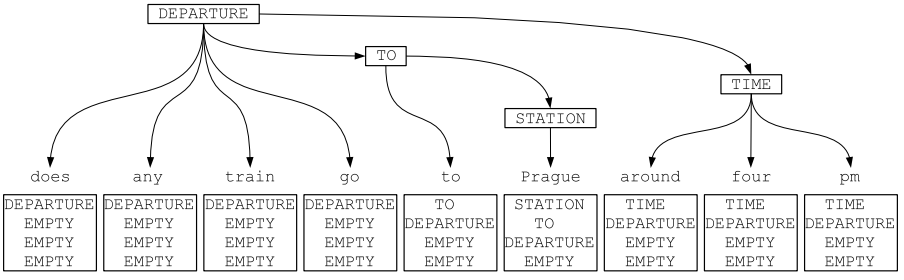

Fig. 1. An example of a full semantic parse tree with the corresponding stack sequence

The transitions between vector states are modeled by stack operations: popping 0 to 3 concepts from a stack, pushing a new concept onto a stack, and generating a word. The first two operations belong to the semantic model which is dgiven by:

$$P(S) = \prod_{t=0}^{T+1} P(n_t|c_{t-1}[1,4]) \cdot \\ \cdot P(c_t[1]|c_t[2,4]) \tag{2}$$

where n_t is the vector state shift operation and takes values in range $0, \ldots, 4$, and c_t at word position t is a vector state of 4 concepts, i.e. $c_t = [c_t[1], c_t[2], c_t[2], c_t[4]]$, where $c_t[1]$ is a preterminal concept dominating the word w_t and $c_t[4]$ is a root concept. The probability $P(n_t|c_{t-1}[1,4])$ represents a model for popping 0 to 3 concepts from a stack. The variable n_t defines the number of concepts which will be popped of a stack. If $n_t = 0$, it relates to growing a stack by one concept. If $n_t = 1$, it relates to replacing preterminal concept $c_t[1]$ by a new concept. If $n_t > 1$, it relates to popping n_t concepts and pushing a new concept. For example, the transition from the vector state represented by the seventh block in Figure 1 is made by popping two concepts TO and STATION and pushing a new concept TIME ($n_6 = 2$). The probability $P(c_t[1]|c_t[2,4])$ represents a model for pushing a new concept $c_t[1]$ onto a stack. The concept $c_t[1]$ is given the rest of a stack $c_t[2,4]$.