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.

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].

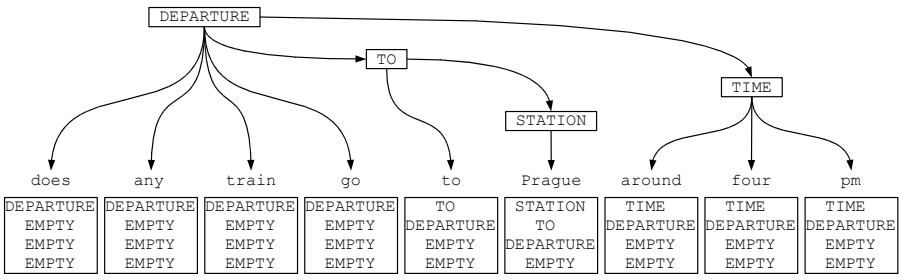

Fig. 1. An example of a full semantic parse tree with the corresponding stack sequence.

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 $c_t[2,4]$.

The lexical model performs the last operation, generation of a word. The lexical model defined as

$$P(W|S) = \prod_{t=0}^{T+1} P(w_t|c_t[1,4]) \tag{3}$$

The word w_t is given $c_t[1,4]$. For more details about the HVS model see [1].

2.1 Training data

The HVS model is possible to train using simple semantics. For the sentence *"Does any train go to Prague around four pm?"*, the corresponding semantics is DEPARTURE(TO(STATION), TIME). Dialogue annotators have to define semantics that represents each utterance, but they need not provide a full parse tree[3]. In Fig. 1, you can see a full parse tree of the example above. To train the HVS model, we eased the Czech human-human train timetable (HHTT) dialogue corpus version DEC-2005. The corpus was described in [2].

Table 1. *A sample dialogue with semantic annotation (a literal translation from Czech to English).*

Speaker	Semantics	Literal English translation
operator	GREETING	the information please
user	GREETING	hello
	DEPARTURE(TIME, TRAIN_TYPE, TO(STATION))	I have a question how can I go today by regional train to <*station*> staryho plzence </*station*>
operator	OTHER_INFO	well we do not have many connections here
	TIME, TIME	now one goes at eight sixteen if you catch it after that only at eleven ten
user	TIME	at eleven ten
	ACCEPT(TIME, FROM(STATION))	it is not so bad at eleven ten from <*station*> hlavniho </*station*> yeah
	CLOSING	yeah well OK

[3] A full parse tree defines not only a tree structure but also alignment of words to leafs of the tree.

2.2 Baseline model

We divided training of the HVS model into three parts: 1) initialization of the semantic and lexical models, 2) estimation of the semantic and lexical models, 3) smoothing of the semantic and lexical models.

We initialized probabilities $P(w_t|c_t[1,4])$, $P(n_t|c_{t-1}[1,4])$, and $P(c_t[1]|c_t[2,4])$ uniformly. In the case of the lexical model, we wanted probability of a word w given a vector state $c[1,4]$ to be the same for all words ($P(w|c[1,4]) = 1/|V| \ \forall w \in V$, where V was a word lexicon).

To estimate the semantic and lexical models, we used the expectation-maximization (EM) algorithm because HHTT corpus did not provide fully annotated tree-bank data. We implemented the linear interpolation smoothing [3] into our model. We smoothed all three probabilities $P(n_t|c_{t-1}[1,4])$, $P(c_t[1]|c_t[2,4])$, and $P(w_t|c_t[1,4])$.

3 Initialization of the lexical model

In comparison with section 2.2, we propose a different initialization based on using negative examples which are collected automatically from a corpus. First, we describe positive examples. Second, we detail negative examples and their automatic extraction from a corpus. Futher, we describe suitable utterances for negative examples extraction. Finally, we present the novel initialization of the lexical model.

3.1 Positive examples

A positive example is a pair of a word and a vector state. A positive example says a word w is possible to generate by a vector state $c[1,4]$. For the utterance from the Fig. 1 *"Does any train go to Prague around four pm?"* with the semantic annotation DEPARTURE(TO(STATION), TIME), a positive example could be a pair (*Prague*, [STATION, TO, DEPARTURE, EMPTY]). Because we do not have full semantic parse trees with vector states aligned to words, the utterance with its semantic annotation contains others positive examples with the word *Prague*, for instance (*Prague*,[TIME, DEPARTURE, EMPTY, EMPTY]).

To determine a probability of the word *Prague* given $c[1,4]$, the EM algorithm must estimate probability over all possible alignments of words in an utterance with a semantic annotation. However, we could make it easier for the EM algorithm if we knew, for example, that the vector state [TIME, DEPARTURE, EMPTY, EMPTY] cannot generate word like *Prague*. As a result, the EM algorithm would have less possible alignments.

3.2 Negative examples

A negative example, similarly to a positive example, is a pair of a word and a vector state. However, a negative example says a word w is *not* possible to

generate by a vector state $c[1,4]$. In other words, negative examples are pairs of words and vector states $c[1,4]$ that do not appear in the same utterances. For instance, the utterance *"Does any train go at five pm?"* with semantic annotation DEPARTURE(TIME) implies that the word *go* is not generated by a vector state different to [STATION, TO, DEPARTURE, EMPTY]. As a result, the pair (*go*, [STATION, TO, DEPARTURE, EMPTY]) could be a negative example.

We analyzed 38 concepts in the corpus, and we found four concepts suitable for negative examples extraction : STATION, TRAIN_TYPE, NUMBER, and TIME. In other words, we search for pairs of any word and a concept form STATION, TRAIN_TYPE, NUMBER, and TIME.

3.3 Selection of utterances for negative examples extraction

Not all utterances are ideal for extraction of negative examples, for instance the utterance *"Weather is pleasant in Prague today."* with semantic annotation OTHER_INFO. If we used the utterance for extracting negative examples, we would have to conclude that the word *Prague* cannot be generated by the vector state [STATION, ...] because the concept OTHER_INFO does not dominate the concept STATION.

To avoid selecting improper utterances for negative examples extraction, we use only utterances containing concepts: ACCEPT, ARRIVAL, DELAY, DEPARTURE, DISTANCE, DURATION, PLATFORM, PRICE, and REJECT because these concepts can dominate concepts STATION, TRAIN_TYPE, NUMBER, and TIME. For example, from the utterance *"What is price of a ticket from Prague at six pm"* with semantics PRICE(FROM(STATION), TIME), we can induce that in the utterance there are not words representing concept TRAIN_TYPE. The algorithm in Fig. 2, finally, describes the process of collecting negative examples from a corpus. For more details about mentioned concepts see [2].

3.4 Application of negative examples

Because negative examples say that a word cannot be generated by a concept, we modify the initialization of the lexical model to utilize negative examples. We still initialize the lexical model uniformly; however, at the same time, we *disable* generation of some words according to collected negative examples. See following equations:

$$x(w, c[1,4]) = \begin{cases} \epsilon & \text{if } (w, c[1,4]) \text{ is a negative example,} \\ 1/|V| & \text{otherwise} \end{cases}$$

$$P(w|c[1,4]) = \frac{x(w, c[1,4])}{\sum_{\overline{w} \in V} x(\overline{w}, c[1,4])} \quad \forall c[1,4] \quad \forall w \in V \tag{4}$$

where ϵ is enough small value, V is a word lexicon.

```
concepts = [ACCEPT, ARRIVAL, DELAY, DEPARTURE, DISTANCE,
            DURATION, PLATFORM, PRICE, REJECT]

dominatedConcepts = [STATION, TRAIN_TYPE, NUMBER, TIME]

for every utterance in trainingSet:
  if utterance.semantics contains a concept from concepts:
    # use the utterance as a source of negative examples

    for every concept in utterance.semantics:
      if concept in dominatedConcepts:
        # use words as negative examples
        negativeExamples[concept].append(utterance.words)
```

Fig. 2. Algorithm to extract negative examples.

We found that it is better to use non-zero value for ϵ because negative examples are not errorless. For example, we need only one wrongly annotated utterance to generate fatal negative example (*Prague*, [STATION, ...]). Consequently, we want to preserve the ability of the lexical model to generate all words because the EM algorithm can overcome a wrong negative example.

4 Experiments

We tested our model on semantic annotations from HHTT corpus version DEC-2005. Currently, the corpus consists of 862 dialogues completely annotated with semantic annotation. Both operators and users are annotated. The corpus has 13769 utterances in total. The vocabulary size is 2667 words. There are 38 semantic concepts in the corpus. In our experiments, the dialogues were randomly divided into training data (619 dialogues - 9928 utterances, 72%), development data (69 dialogues - 1108 utterances, 8%), test data (174 dialogues - 2733 utterances, 20%).

We evaluated our experiments using the standard PARSEVAL measures [4]; labeled precision (LP), recall (LR), and labeled F-measure (LF), which are computed as follows:

$$LP = \frac{\#\ of\ correct\ concepts\ in\ P}{\#\ of\ concepts\ \ in\ P} \cdot 100\%$$

$$LR = \frac{\#\ of\ correct\ concepts\ in\ P}{\#\ of\ concepts\ in\ T} \cdot 100\%$$

$$LF = \frac{2 \cdot LP \cdot LR}{LP + LR} \cdot 100\%$$

where P is the candidate parse tree (our estimated the most propable parse), T is the corresponding correct parse tree from the corpus. A concept in P is *correct* if

there exists a concept in T of the same label that spans the same words. We used the *evalb* program from Satoshi Sekine and Michael John Collins[4] to compute these scores. We tested whether the observed differences in PARSEVAL measures are significant at $p = 0.01$ using a stratified shuffling test with one million trials from Dan Bikel[5].

4.1 Validation

To determine the effect of negative examples on the initialization of the lexical model, we evaluated different initializations of the lexical model on our development data. We evaluated the effect separately for each concept NUMBER, TIME, STATION, and TRAIN_TYPE. Table 2 compares the results of the baseline system with uniformly initialized lexical model.

Table 2. PARSEVAL scores on the development data.

	LP	LR	LF	p-value
baseline	43.7	51.3	47.2	
NUMBER	43.6	51.2	47.1	> 0.01
TIME	43.6	45.7	44.6	< 0.01
STATION	46.9	53.8	50.1	< 0.01
TRAIN_TYPE	44.8	51.2	47.6	> 0.01
NST	48.7	54.4	51.4	< 0.001

The use of negative examples for the initialization of the lexical model for the concept NUMBER seems to lower the PARSEVAL scores; however, the difference was not statistically significant.

The use of negative examples for the initialization of the lexical model for the concept TIME significantly lower the PARSEVAL scores. As a result, the concept TIME was excluded from futher experiments.

The use of negative examples for the initialization of the lexical model for the concept STATION significantly improves the PARSEVAL scores.

The use of negative examples for the initialization of the lexical model for the concept TRAIN_TYPE seems to improve the PARSEVAL scores; however, the difference was not statistically significant.

Finally, we tested the initialization of the lexical model using the concepts NUMBER, STATION, and TRAIN_TYPE together (NST) (see Table 2, the row NST). These results are statistically significant on the development data.

4.2 Test data results

Table 3 shows the results of the baseline initialization of the lexical model using uniform initialization and the initialization of the lexical model using negative

[4] http://nlp.cs.nyu.edu/evalb
[5] http://www.cis.upenn.edu/∼dbikel/software.html

examples for the concepts NUMBER, STATION, and TRAIN_TYPE (NST). We report results for both test and development data. The use of negative examples for the initialization of the lexical model for the concepts NUMBER, TRAIN, and TRAIN_TYPE significantly improve PARSEVAL scores.

Table 3. PARSEVAL scores on the test and the development data.

	Test data				Development data			
	LP	LR	LF	p-value	LP	LR	LF	p-value
baseline	42.2	49.2	45.4		43.7	51.3	47.2	
NST	46.6	51.9	49.1	< 0.001	48.7	54.4	51.4	< 0.001

5 Conclusion

This paper has presented the use of negative examples in training the HVS model. We identified suitable utterances for extraction of negative examples. We found three concepts NUMBER, STATION, and TRAIN_TYPE for which the initialization using negative examples of the lexical model significantly improves the PARSEVAL scores. The labeled F-measure (LF) was increased from 45.4% to 49.1%.

In the future work, we want futher exploit different ways of initialization of the lexical model.

Acknowledgment

This work was supported by the Ministry of Education of the Czech Republic under project 1M0567.

References

1. He, Y., Young, S.: Semantic processing using the hidden vector state model. Computer Speech and Language **19:1** (2005) 85–106
2. Jurcicek, F., Zahradil, J., Jelinek, L.: A human-human train timetable dialogue corpus. In: Proceedings of 9th European Conference on Speech Communication and Technology, Lisboa, Portugal (2005)
3. Jelinek, F.: Statistical methods for speech recognition. MIT Press, Cambridge, MA, USA (1997)
4. Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Strzalkowski, S.T.: A procedure for quantitatively comparing the syntactic coverage of english grammars. In: Proceedings of the 1990 DARPA Speech and Natural Language Workshop, Pacific Grove, CA (1991) 306–311