HIERARCHICAL DISCRIMINATIVE MODEL FOR SPOKEN LANGUAGE UNDERSTANDING

Jan Švec[1], *Luboš Šmídl*[2], *Pavel Ircing*[1]

[1]Department of Cybernetics, [2]NTIS - New Technologies for Information Society
University of West Bohemia
Univerzitní 22, Pilsen, Czech Republic
{honzas,smidl,ircing}@kky.zcu.cz

ABSTRACT

The paper presents a new discriminative model for statistical spoken language understanding designed for use in spoken dialog systems. The parsing algorithm uses lexicalized grammar derived from unaligned training data with probability estimates generated by multiclass classifiers. The generated semantic trees are partially aligned with the input sentence to provide lexical realisation of semantic concepts. The model was evaluated on two semantically annotated corpora and in both tasks it outperforms the baseline Hidden Vector State parser and Semantic Tuple Classifiers model. The experiments were performed using both transcribed data and recognized lattices. The innovative aspect of using phoneme lattices in the understanding process instead of word lattices is examined and described.

***Index Terms*—** Automatic speech recognition, Dialogue systems, Spoken language understanding

1. INTRODUCTION

The spoken language understanding (SLU) module is a crucial component of a dialog system. If the recognition and understanding performance is poor, the whole dialog could consist of many misunderstandings and the dialog system would be unusable. The statistical based SLU systems were widely studied in the past decade. One of the possible solutions is a lexicalized statistical parser introduced in [1]. It is based on the lexicalized PCFG with rule probabilities derived from aligned data. The new generative model – Hidden Vector State (HVS) parser – was introduced in [2]. It allows to train the SLU from an unaligned semantic tree annotation. Although the class of generated semantic trees of the original HVS parser is limited to left-branching trees it could be extended to parse also a left-right-branching trees [3]. The left-right-branching algorithm improves the performance over the HVS parser [4]. Although the results seemed to be promising, the overall performance of pipelined ASR and SLU system was not ideal. The performance of HVS parser was outperformed on the ATIS task [5] by very simple discriminative model called Semantic Tuple Classifiers (STC) which is based on a set of binary SVM classifiers predicting parts of the output semantic tree [6].

2. HIERARCHICAL DISCRIMINATIVE MODEL

The work described in this paper was motivated by the discriminative model such as STC. Instead of using the output heuristics to reconstruct the semantic tree, the described Hierarchical Discriminative Model (HDM) uses a semantic grammar derived from training data with expansion probabilities predicted by discriminative classifiers. We use a terminology from artificial neural networks. The described model can consist of two or three layers connected in a feed-forward manner. Therefore we use the terms input, hidden, and output layer (Fig. 1).

2.1. Output layer

The HDM output layer uses a semantic grammar similar to lexicalized probabilistic context-free grammars. The parsing algorithm generates an unordered semantic tree which is only partially aligned with the underlaying lexical representation of the user's utterance. The semantic grammar is parameterized by the utterance u and consists of a tuple $G_u = (\Theta, R_u, S)$, where Θ is a set of semantic concepts, R_u is a set of grammar rules dependent on the utterance u and $S \in \Theta$ is a root concept (starting symbol of the parsing algorithm). The rules R_u are in the form $A \to \beta\ [p]$, where $A \in \Theta$, $\beta \subseteq \{\nu\} \cup \Theta$, ν is a special symbol representing rules without lexical realisation in the utterance u (see below) and p is the probability of concept A having a set of child nodes β:

$$p = P(A \to \beta|u) = P(\beta|A, u) \qquad (1)$$

The right-hand side of the rule $A \to \beta\ [p]$ is an unordered set, therefore the generated semantic tree is unordered – there is no ordering relation between the child nodes. There is also no distinction between terminal and non-terminal symbols, all symbols from Θ are supposed to be non-terminals, i.e. any $A \in \Theta$ can occur on a left-hand side of some rule. Due to missing terminal symbols, a correspondence of semantic tree and the underlaying lexical representation is given by the rules of the form $A \to \emptyset\ [p]$ or $A \to \{\nu\}\ [p]$. The first form represents the case in which the semantic concept A has a lexical representation in the utterance u (with probability p). On the other hand the concept expanded to $\{\nu\}$ does not have a lexical representation in the utterance u and the symbol A is not part of the semantic tree (with probability p).

Construction of the semantic grammar is relatively simple – first, the non-parameterised set of rules R is generated from training data. The set R contains expansions of each concept A seen during the training phase. In addition the expansions $A \to \emptyset$ and $A \to \{\nu\}$ are added into R for each $A \in \Theta$. We will denote the set of all possible expansion of the concept A as $B_A = \{\beta : A \to \beta \in R\}$. We use a special starting symbol S_0 to model the top-level concepts of semantic trees. For each tree the symbol S_0 is a parent of all top-level concepts in an annotated tree. For example the semantic tree TIME, TO(STATION) leads to the following set of rules R: $S_0 \to \{\text{TIME, TO}\}$; TO $\to$ STATION; TIME $\to \emptyset$ and STATION $\to \emptyset$.

The probabilities $P(A \to \beta|u)$ are estimated using multiclass discriminative models. In this work we used Support Vector Machines (SVM) with RBF kernel but arbitrary multiclass classifier providing posterior estimates can be used. To obtain probability distribution over the target classes the decision margin of an SVM classifier is mapped to probability using approach described in [7]. To train the classifiers, the annotated semantic tree s_i has to be transformed into a target classes $t_A(s_i)$ for each classifier predicting the expansions of concept A. For each concept A in s_i the target class is $t_A(s_i) = (A \to \beta)$ if A has children β in the tree s_i. Otherwise A is a leaf concept and $t_A(s_i) = (A \to \emptyset)$. For concepts not occurring in s_i we use $t_A(s_i) = (A \to \{\nu\})$. Then the output classifier for each $A \in \Theta$ is trained using input feature vector $\mathbf{d}(u_i)$ and target class $t_A(s_i)$. In the current implementation we use one SVM multiclass classifier with RBF kernel for each $A \in \Theta$. Once the classifiers are trained it is possible to predict the posterior probabilities $P(t_A|\mathbf{d}(u)) = P(A \to \beta|u)$.

Given the parse tree π and the rule probabilities, we are able to define the overall tree probability $P(\pi|u)$ conditioned on the utterance u as a product of probabilities of rules occurring in the tree:

$$P(\pi|u) = \prod_{A \to \beta \in \pi} P(A \to \beta|u) \tag{2}$$

The goal of the parsing algorithm is to find the most probable semantic tree corresponding to the utterance u. First the multiclass classifier for each concept $A \in \Theta$ predicts the probability distribution over the corresponding sets B_A. Then the set of generic rules R is extended with probabilities $P(A \to \beta|u)$ to form the set R_u and the semantic grammar $G_u = (\Theta, R_u, S_0)$. The algorithm iteratively expands the nodes of the semantic tree starting with the root symbol S_0 and using the rules from R_u. The algorithm is a standard best-first search (BFS) algorithm with the cost defined as a negative logarithm of partial tree probability given by Eq. 2. We represent the partial tree r as a list of applied rules. For each A we will denote the number of times the concept A occurs on left-hand side of all rules in r as $L_A(r)$. The number $R_A(r)$ denotes the number of times A occurs on right-hand side of all rules in r. The BFS algorithm stops if the best partial tree r matches the condition $L_A(r) = R_A(r)\ \forall A \in \Theta$ and there is no rule of type $A \to \{\nu\}$ in r. The parsing algorithm also allows to find the n-th most probable semantic tree simply by continuing the BFS after the first semantic tree is found.

To avoid possibly infinite algorithm, the recursive rules containing the concept A in the set B_A are not allowed. Even the indirect recurrence is undesirable. This condition is not limiting in majority of applications as discussed in Section 2.4.

The partial alignment of an utterance u and the semantic tree π is done by determining the lexical representation of terminal nodes. It is not necessary to find the lexical representation for every node in the tree because in the spoken dialog system the presence of some concept is sufficient – there is no need to know the exact lexical realisation. An example of such concept from the HHTT corpus (Sec. 3) is DEP (departure) – the dialog manager only needs to know the probability of occurrence of this concept in the tree, not the corresponding realisation.

In the current implementation of HDM, the approach of substituting lexical classes with corresponding class labels is used [6]. The words that match the predefined list of lexical realisations are replaced with the class labels and the correspondence between class labels and leaf concepts is defined. For example the words "half past nine" are replaced with a class label *time* and aligned with a concept TIME. This approach allows the HDM to output semantic trees without lexical realisation of some concepts – for example the semantic tree can contain the concept STATION but this concept would not be linked with any uttered word because the uttered sentence does not contain the corresponding lexical class. This could occur in the case the user uses an out-of-vocabulary word which is misrecognized by an ASR but the output layer generates a tree with the STATION concept based on the surrounding words. This information can be used in the dialog strategy to inform an user and instruct him to reformulate the utterance.

2.2. Input layer

The input feature vector for each classifier can be formed from lexical-syntactic features such as a frequency of word n-grams [6]. In this work we use rational kernels theory with the counting transducer to represent the input lattice with feature vector composed from expected counts of n-grams $n = 1, 2, \dots n_{max}$ [8, 9]. The advantage of rational kernels is the direct computation of kernel functions $K(u_i, u_j)$ between two utterances u_i, u_j represented with weighted finite state transducers (WFSTs) without the need to explicitly represent the feature vector. In addition we use feature space kernel normalization defined in [10] to normalize values of kernels to an interval $[0, 1]$. The normalized values $K(u_i, u_j)$ are used directly in the SVM classifiers (see Eq. 3). In the hierarchical discriminative model the part computing the kernel values is called an *input layer*.

By using WFSTs to represent utterances we are able to train an understanding model which takes into account the uncertainty of ASR lattices. The HDM is not limited to use just word-lattices but we can use arbitrary acyclic WFST such as word-strings (for 1-best hypothesis) or phoneme lattices. The input layer was implemented using the OpenFST library [11] which provides excellent computing performance. On a laptop with Intel i7 2.4 GHz processor we were able to compute the kernel function between one unseen phoneme lattice and 5.4k training phoneme lattices on a single processor core in approximately 20ms and for the word lattices in about 1.5 ms.

2.3. Hidden layer

Although the hierarchical discriminative model is able to process directly the input lattices represented by the kernel functions computed in the input layer, our experiments showed that by using some input preprocessing the understanding performance can be significantly improved. The preprocessing is performed in a *hidden layer*. In the current implementation we use the Semantic Tuple Classifiers (STC) model [6] to reduce the input space from thousands of possible n-grams into a much smaller set of semantic tuples. The STC uses a set of binary classifiers trained to predict the presence or absence of a *semantic tuple*. The semantic tuple is defined as subsequence of path from the semantic tree root node to any other node, for example the semantic tree DEP(TIME, TO(STATION)) contains following semantic tuples: DEP-TIME, DEP-TO, DEP-TO-STATION, TO-STATION and four trivial tuples: DEP, TIME, TO, STATION. The original approach by Mairesse et al. [6] used a heuristic to reconstruct the semantic tree from the set of predicted semantic tuples. In case of using the STC as a hidden layer of HDM, the output heuristic is replaced with the HDM parsing algorithm and the tree reconstruction is based on a classifiers trained from annotated data.

The STC hidden layer uses exactly one classifier of each semantic tuple $\mathbf{t} \in S_N$. The set S_N consists of all semantic tuples occurring in training set more than N-times. The hidden layer is used to transform the input utterance u to a feature vector $\mathbf{d}(u) = [d_{\mathbf{t}}(u)]\ \mathbf{t} \in S_N$ given the set of k training utterances $\{u_i\}_{i=1}^{k}$. The

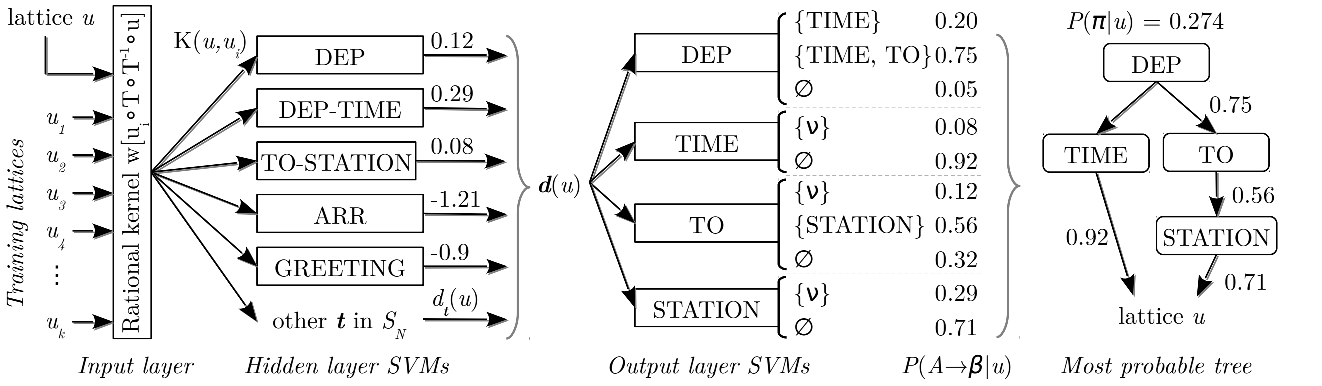

Fig. 1: HDM schema. Output layer classifiers for the remaining concepts $A \in \Theta$ were omitted for clarity and the expansions were simplified.

value $d_{\mathbf{t}}(u)$ is a distance to a decision boundary (i.e. decision function) of the SVM classifier corresponding to tuple $\mathbf{t}$ given the input u:

$$d_{\mathbf{t}}(u) = \sum_{i=1}^{k} \alpha_i^{\mathbf{t}} y_i^{\mathbf{t}} K(u, u_i) + b^{\mathbf{t}} \tag{3}$$

where $\alpha_i^{\mathbf{t}}, b^{\mathbf{t}}$ are parameters of SVM corresponding to a tuple $\mathbf{t}$, $y_i^{\mathbf{t}}$ is +1 if the tuple $\mathbf{t}$ is present in the annotated semantic tree of utterance u_i, -1 otherwise and $K(u, u_i)$ is the kernel function between u and u_i computed by the input layer. Then the output layer is trained from the feature vectors $d(u_i), i = 1, \ldots, k$ and similarly the prediction of semantic tree for an unseen utterance u is done first by predicting $d(u)$ and then by using the output layer SVMs to predict the probability distributions $P(A \rightarrow \beta|u)$. The trained three layer HDM contains $|S_N|$ binary SVM classifiers in the hidden layer (one classifier for each semantic tuple) and $|\Theta|$ multiclass SVM classifier in the output layer (one classifier for each semantic concept).

2.4. Algorithm limitations

In this section we will briefly discuss the limitations of the HDM defined in the previous section. The first limitation arise from the definition of semantic grammar rules – $A \rightarrow \beta$ where β is an unordered set and also the generated semantic trees are unordered. This does not limit the dialog manager in majority of domains because the dialog strategy usually does not rely on the concept ordering. This also implies that some node cannot have multiple direct children with the same concept, e.g. semantic annotation DEP(TIME, TO(STATION), TIME). To overcome this limitation we recommend depending on the dialog manager implementation: (a) to modify the semantic annotation to contain numbered concepts, eg. DEP(TIME-1, TO(STATION), TIME-2), or (b) to unify the multiple concepts into the annotation DEP(TIME, TO(STATION)) and then to assign two lexical realisations of entity time to single concept TIME.

The second limitation comes from the parsing algorithm – the semantic grammar is not allowed to generate semantic tree with a node labeled with concept A and at the same time recursively containing concept A in a subtree of A. This is not very limiting because the utterances in common dialogs do not have recursive structure. Therefore the recursive rules can be disallowed by the annotation schema without the loss of generality.

The last limitation is imposed by the probabilities assigned to the rules. Each occurrence of some rule from R_u shares the assigned probability with each other. For example in the semantic tree DEP(FROM(STATION), TO(STATION)) the probability of STATION $\rightarrow \emptyset$ (the probability of STATION being the leaf node of semantic tree) is the same for both nodes containing the STATION concept. Nevertheless the probabilities of subtrees FROM(STATION) and TO(STATIONS) are generally different because the probabilities of rules FROM $\rightarrow$ STATION and TO $\rightarrow$ STATION are different.

3. RESULTS

We used two semantically annotated corpora designed for spoken language understanding in a spontaneous dialog. The first one was the Human-Human Train-Timetable (HHTT) corpus [4] used in our previous work on HVS parser [3, 12]. The corpus contains inquiries and answers about train connections. The second one was a newly collected Czech Intelligent Telephone Assistant (TIA) corpus containing utterances about meeting planning, corporate resource sharing and conference call management. These corpora contain unaligned semantic trees together with word-level transcriptions. We have split the corpora into train, development and test data sets (72:8:20) at the dialog level, so that the speakers do not overlap.

	HHTT	TIA
$\|\Theta\|$ (# different concepts)	28	24
$\|S_N\|$	70	83
# train sentences (concepts)	5240 (8849)	4337 (9027)
# devel. sentences (concepts)	570 (989)	469 (1107)
# test sentences (concepts)	1439 (2546)	1073 (2305)
ASR Vocabulary size	13886	2624
Word ASR *Acc* (OOV rate)	72.9% (7.5%)	63.0% (8.6%)
Phoneme ASR *Acc*	74.3%	67.5%

Table 1: Corpora characteristics.

To evaluate the SLU performance we used the *concept accuracy* measure defined as $cAcc = \frac{H-S-D-I}{N}$ where H is the number of correctly recognized concepts, N is the number of concepts in reference and S, D, I are the numbers of substitutions, deletions and insertions [3]. The reference and hypothesis tree were aligned using algorithm described in [13]. Since the corpora do not contain any tree alignment only the parsed concepts and structure of semantic tree were evaluated. We used our in-house LVCSR decoder to obtain the word and phoneme lattices [14]. The word language model used to recognize HHTT data was trigram class-based LM while the TIA data was recognized using simple trigram LM. To obtain the

	HHTT	*cAcc*[%]		TIA	*cAcc*[%]	
Model	*trans*	*1best*	*latt*	*trans*	*1best*	*latt*
HVS	74.9	64.5	–	–	–	–
STC	76.0	66.7	67.9	74.7	65.8	65.6
HDM-2	78.0	67.0	68.3	78.4	70.3	69.4
HDM-3	80.5	70.4	72.8	80.9	72.7	73.9

Table 2: Results for different word-level models.

	HHTT	*cAcc*[%]		TIA	*cAcc*[%]	
Model	*ph-ad*	*ph*	*map*	*ph-ad*	*ph*	*map*
STC	58.9	62.7	69.5	61.6	65.2	69.8
HDM-2	61.6	63.2	70.7	64.5	67.6	73.0
HDM-3	67.1	69.2	74.8	69.1	70.9	75.3

Table 3: Results for different phoneme-level models.

phoneme lattices the 5-gram phoneme language model was used. The recognition accuracy and other characteristics of both corpora are summarized in Tab. 1.

We performed a number of experiments to compare the performance of baseline HVS and STC models and the described HDM. We used two variants of HDM - two-layer structure with the hidden layer omitted (HDM-2) and three-layer with STC as a hidden layer (HDM-3). The models were trained without any lexical class detection in the input layer – this fact negatively influenced mainly the STC model where the lexical classes are beneficial. We decided not use lexical classes because then we are able to directly compare the HVS, STC and both word-based and phoneme based HDM. In the word-level experiments we used n-gram rational kernel with $n_{max} = 3$, for the phoneme-level experiments we used $n_{max} = 5$. The parameter N of the set of semantic tuples S_N was set to 30. These parameters were tuned to optimize the performance on the development data. No contextual information (eg. previous utterances, semantics, dialogue state) were used.

The first set of experiments was performed using word-level transcriptions (Tab. 2, *trans*). The baseline models were HVS and STC models. Both the two-layer and three-layer hierarchical models outperformed the baseline while the HDM-3 performed better then the HDM-2. Then second set of word-level experiments was performed using ASR outputs (Tab. 2). First we used the word recognizer and we evaluated the increase of concept accuracy while parsing the one-best ASR hypothesis (*1best*) and the word lattice (*latt*). In the HHTT task, the increase was consistently about 1% absolutely for STC and HDM-2 while in the TIA task there was no difference between one-best a n-best results. The slightly bigger difference was encountered for the HDM-3 model.

Then we focused on a new approach to speech understanding – SLU from subword units, especially from phonemes. By using the input layer we are able to quickly evaluate the kernel between the unseen phoneme lattice and the training set. The use of phonemes has a number of advantages – first it is not necessary to train and tune the word-level language model (LM). The phoneme LM can be easily adapted from a generic one because in a given language there is only a limited number of phonemes. Therefore we can use the generic phoneme LM to recognize the unannotated training data and then train the adapted phoneme LM from the recognized sequence of phonemes. The second advantage is that the phoneme-level n-gram rational kernel substitutes lemmatization or stemming of input because the kernel matches also substrings of words. To demonstrate this we included the results on phoneme lattices. First we used the phoneme LM trained directly from the forced-alignments of transcribed training data (Tab. 3 *ph*). The second set of experiments was generated using an adapted phoneme LM (Tab. 3, *ph-ad*). There is a stable tolerable decline of 2% of $cAcc$ by using the adapted phoneme LM. We also performed an experiment with the word lattices mapped into phonemes according to the pronunciation dictionary (Tab. 3, *map*). In this case the performance increased by 5% of $cAcc$ and outperformed the word-level HDM results.

4. CONCLUSION AND FUTURE WORK

The HDM with hidden layer outperforms both baseline models and also the two-layer HDM. The transformation using the hidden layer allows to better parameterize the utterance and the prediction of output layer incorporate the knowledge about all semantic concepts (more precisely about all semantic tuples from S_N). This allows to easily detect the out-of-topic (OOT) utterances, for example the TIA test set contain 210 OOT utterance and the OOT concept is detected with precision 68.2% and recall 75.7% (F=71.8%, ASR word-level lattices). The results for word-level ASR system do not show any significant difference between the 1-best and full lattice HDM results. This could be caused by a low-quality posterior probabilities from an ASR system. In the future we plan to experiment with more sophisticated methods for generating ASR lattices, such as [15].

The results obtained for phoneme-level HDM are comparable to word-level HDM. The results for lattices obtained by using an adapted phoneme-level LM suggest that the SLU model can be developed very quickly without the need to collect and transcribe a large number of sentences to train a robust word-level LM. On the the other hand, if the word-level LM is available the mapping of word lattices to phonemes can significantly improve the understanding performance. In the future research, we would like to devote to the lexical classes detection in phoneme-lattices and use the information about their occurrences to support the decisions in the output layer. Our preliminary experiments showed that by adding the scores of lexical classes into a feature vector $d(u)$, we are able to mitigate the difference between an adopted and trained phoneme LM.

The HDM is able to process a wide range of different inputs starting with word strings and continuing with word and phoneme lattices. The results also showed that the SLU from phonemes is possible and we are able to simplify the SLU process by eliminating the word-level LM, which is expensive to obtain, hard to adapt and limited in the sense of its lexicon. The results obtained using word lattices mapped to phonemes are very promising. In the future research, we want to focus also on the fusion of phoneme recognizer (which is able to handle OOV words and non-standard pronunciations) with word recognizer (able to model longer inter-word dependencies). Summarizing the parsing performance on recognized data the HDM was able to increase the concept accuracy on the HHTT task from 66.7% to 74.8% and on the TIA task from 65.8% to 75.3% in comparison with the STC model. In addition the HDM is able to detect out-of-topic sentences and to use phoneme-level information in the parsing process.

5. ACKNOWLEDGEMENT

This research was supported by the Technology Agency of the Czech Republic, project No. TE01020197 and by the European Regional Development Fund (ERDF), project “New Technologies for Information Society” (NTIS), European Centre of Excellence, ED1.1.00/02.0090.

6. REFERENCES

[1] Eugene Charniak, “Immediate-Head Parsing for Language Models,” in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, Toulouse, France, 2001, number 3, pp. 124–131, ACM.

[2] Yulan He and Steve Young, “Hidden vector state model for hierarchical semantic parsing,” *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003.1*, vol. 1, pp. 268–271, 2003.

[3] Filip Jurčíček, Jan Švec, and Luděk Müller, “Extension of HVS semantic parser by allowing left-right branching,” in *IEEE International Conference on Acoustics Speed and Signal Processing*, 2008, number 1, pp. 4993–4996.

[4] Filip Jurčíček, Jiří Zahradil, and Libor Jelínek, “A human-human train timetable dialogue corpus,” *Proceedings of EUROSPEECH, Lisboa*, pp. 1525–1528, 2005.

[5] Deborah A. Dah, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriber, “Expanding the scope of the ATIS task: The ATIS-3 corpus,” in *Proceedings of the workshop on Human Language Technology*, Stroudsburg, 1994, pp. 43–48.

[6] François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young, “Spoken language understanding from unaligned data using discriminative classification models,” in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009.*, Taipei, 2009, pp. 4749–4752, IEEE.

[7] Ting-Fan Wu, Chih-jen Lin, and Ruby C. Weng, “Probability estimates for multi-class classification by pairwise coupling,” *The Journal of Machine Learning Research*, vol. 5, pp. 975–1005, 2004.

[8] Corinna Cortes, Patrick Haffner, and Mehryar Mohri, “Rational kernels: Theory and algorithms,” *The Journal of Machine Learning*, vol. 5, pp. 1035–1062, 2004.

[9] Jan Švec and Pavel Ircing, “Efficient algorithm for rational kernel evaluation in large lattice sets,” in *IEEE International Conference on Acoustics Speech and Signal Processing*, Vancouver, Canada, 2013, IEEE.

[10] Arnulf B.A. Graf, Alexander J. Smola, and Silvio Borer, “Classification in a normalized feature space using support vector machines.,” *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 597–605, Jan. 2003.

[11] Cyril Allauzen, Michael Riley, and Johan Schalkwyk, “OpenFst: A general and efficient weighted finite-state transducer library,” *Implementation and Application of Automata*, vol. 4783, pp. 11–23, 2007.

[12] Jan Švec and Filip Jurčíček, “Extended Hidden Vector State Parser,” *Text, Speech and Dialogue*, vol. 5729, pp. 403–410, 2009.

[13] Kaizhong Zhang, “A Constrained Edit Distance Between Unordered Labeled Trees,” *Algorithmica*, vol. 15, no. 3, pp. 205–222, 1996.

[14] Aleš Pražák, Josef V. Psutka, Jan Hoidekr, Jakub Kanis, Luděk Müller, and Josef Psutka, “Automatic online subtitling of the Czech parliament meetings,” *Text, Speech and Dialogue*, vol. 4188, no. 1, pp. 501–508, 2006.

[15] Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motlíček, Yanmin Qian, Korbinian Riedhammer, Karel Veselý, and Ngoc Thang Vu, “Generating exact lattices in the WFST framework,” in *IEEE International Conference on Acoustics Speech and Signal Processing*, Kyoto, Japan, 2012, vol. 213850, pp. 4213–4216, IEEE.