SEMANTIC ENTITY DETECTION FROM MULTIPLE ASR HYPOTHESES WITHIN THE WFST FRAMEWORK

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

[1]Department of Cybernetics, University of West Bohemia, Pilsen, Czech Republic

[honzas,ircing,smidl]@kky.zcu.cz

ABSTRACT

The paper presents a novel approach to named entity detection from ASR lattices. Since the described method not only detects the named entities but also assigns a detailed semantic interpretation to them, we call our approach the semantic entity detection. All the algorithms are designed to use automata operations defined within the framework of weighted finite state transducers (WFST) – the ASR lattices are nowadays frequently represented as weighted acceptors. The expert knowledge about the semantics of the task at hand can be first expressed in the form of a context free grammar and then converted to the FST form. We use a WFST optimization to obtain compact representation of the ASR lattice. The WFST framework also allows to use the word confusion networks as another representation of multiple ASR hypotheses. That way we can use the full power of composition and optimization operations implemented in the OpenFST toolkit for our semantic entity detection algorithm. The devised method also employs the concept of a factor automaton; this approach allows us to overcome the need for a filler model and consequently makes the method more general. The paper includes experimental evaluation of the proposed algorithm and compares the performance obtained by using the one-best word hypothesis, optimized lattices and word confusion networks.

Index Terms: spoken language understanding, named entity detection, spoken dialog systems

1. INTRODUCTION

The spoken language understanding (SLU) task can be viewed as a problem of assigning a semantic information to a given acoustic observation O. Most often the observation O is processed in a standard large vocabulary automatic speech recognizer to produce a word lattice encoding word hypotheses and associated probability distribution $P(W|O)$.

The meaning can be represented as attribute-values pairs or semantic trees. In our previous research we used the Hierarchical Discriminative Model (HDM) to generate abstract semantic trees which represent the semantic structure of the user's utterance [1]. To employ the HDM model in the dialog system, we need to assign concrete semantic information to the nodes of an abstract semantic tree which is not in general aligned to the lexical realisation of the utterance. Therefore we designed a new algorithm which is able to extract the information about semantic entities in the utterance and consequently link them with the abstract semantic tree to form the full semantics of the utterance.

In this paper we focus only on the semantic entity detection algorithm. This algorithm performs both the detection of presence of some type of semantic entity and the extraction of the semantic entity content. The main requirement on the algorithm was the ability to assign the posterior probabilities to a given sequence of semantic entities. Therefore we define the random variable E over a set of sequences $e = \{e_1, e_2, \ldots\}$ where e_i is i-th *semantic entity*. The described algorithm then computes the semantic entity lattice which represents the distribution $P(E|O)$ based on $P(W|O)$.

The semantic entity is an instance of common "data type" used in speech – date, time, proper name, address, city name etc. The semantic entity consists of *entity type* and *interpretation* representing concrete entity. We join the type and interpretation using the colon, e.g. time *half past seven* can have the following semantic entity: *time:30:m:past:07:h*. Other examples could be *dep_airport:Prague* or *person:John:Smith*. The entity interpretation is used to construct "a machine readable" representation of semantic entity and its syntax depends on the semantic entity type.

The idea of tagging the semantic entities using weighted finite state transducers (WFST) is generally not new. For example, Béchet et al. [2] developed a method for named entity (NE) detection based on WFST. Their system uses the NE tagger which preselects the regions where NE can occur and then applies lattice parsing algorithm to obtain the semantic entity. In contrast to WFST semantic interpretation presented by Raymond et al. [3], our approach does not use any kind of filler model. Only factors of input sequence matching the semantic entity grammars are parsed to speedup the processing.

The proposed algorithm is similar to the lattice indexing method described in [4]. Instead of using a large set of lattices and a simple query, in this paper we use an input in the form of single lattice and a complex query which is generated from a grammar. The complex knowledge representation leads to a set of unambiguous semantic entities and this paper proposes a disambiguation algorithm based on integer linear programming.

There is also another work by Hakani-Tür et al. [5] in which they use word confusions networks (WCNs) to detect semantic entities in ASR output. Their results show that the performance obtained by using the WCNs and word lattices is the same but the WCNs allow higher processing speeds – 25x faster than the lattice-based approach. The WCNs are generally used to represent uncertain ASR output because of their simple structure and small size in comparison with general ASR lattices. In this paper we propose to use the lattices optimized using standard WFST algorithms as a better representation of multiple ASR hypotheses. Since the WCN is virtually also an acyclic WFST, the algorithm presented in this paper is evaluated using both the optimized word lattices and WCNs.

The Section 2 describes the raw and optimized lattices and word confusion networks and states the hypothesis that WCNs are very inaccurate in representing the posterior probabilities of word sequences longer than one word. The Section 3 describes the novelty algorithm for semantic entity detection and an algorithm for constructing the semantic entity lattice. The Section 4 presents the experimental results and finally the Section 5 concludes the paper.

2. WORD CONFUSION NETWORKS AND LATTICES

The common way to represent multiple ASR hypotheses are the word lattices. These raw lattices are a by-product of the Viterbi decoding of the HMM representing the acoustic and language models. In general, the raw lattices contain number of hypotheses and many of them have a negligible probability. Therefore it is common to reduce the size of the lattice prior to further processing. In this paper we used two different reduced structures – the optimized lattices and word confusion networks. In both cases the result is a normalized acyclic WFST in which the sum of probabilities of all hypotheses is one. Such a WFST defines the probability distribution over word hypotheses $P(W|O)$.

We generate the optimized lattices from the raw lattices using the following steps, which were inspired by the paper [4]:

1. Start with the lattice defined over the tropical semiring.
2. If the transition represents a non-speech event, change the label to ϵ.
3. Apply ϵ-removal algorithm.
4. Prune the lattices using threshold t_p.
5. Change the semiring of the lattices to logarithmic semiring.
6. Apply weight-pushing algorithm which ensures that the sum of all transition probabilities from a given state is one.
7. Determinize the lattice, the result of determinization is the *optimized lattice*

To convert the raw lattice to a word confusion network, we used an algorithm described in [5]. The algorithm first computes the posterior probabilities for each transition in the lattice. Then the so called *pivot path* is selected. In this paper we used the best path through a lattice as a pivot. The pivot transitions are weighted using the corresponding posterior probabilities. In the next step, the algorithm iterates over all transitions from the raw lattice. For each transition t labelled with the word w, the two consequent states s_0 and s_1 on the pivot path are selected based on the maximum overlap criterion. If there is already a transition $\bar{t}$ between s_0 and s_1 which is labelled with w, the posterior probability of t is added to $\bar{t}$. If there is no such transition, the new transition labelled with w and weighted by the posterior probability of t is added between s_0 and s_1. The word confusion networks (WCNs) have similar properties as the optimized lattices. The total sum of all hypothesis probabilities is also one. In addition the transition weights directly represent the single word posterior probability. In general the WCNs keep the oracle accuracy of the raw lattice while the optimized lattices have lower oracle accuracy.

The WCNs are often used in spoken language understanding [5, 6]. One of the drawbacks of using WCNs is that they exactly represent just the posterior probability of single words. But in tasks such as SLU the posterior probabilities of longer word sequences are needed. In this case the posterior estimates given by WCNs are not accurate because the consecutive words are modeled as being statistically independent.

To describe the problem, we use the original raw lattice depicted in Fig. 1. It contains three hypotheses ab (posterior probability 0.5), cd (0.3) and ef (0.2). The WCN created from this lattices is depicted in Fig. 2. The WCN has a well-known sausage-like structure with the first segment containing three parallel words a, c and e and the second segment containing b, d and f. The WCN contains nine different hypothesis: ab (posterior probability 0.25), ad (0.15), af (0.1), cb (0.15), cd (0.09), cf (0.06), eb (0.1), ed (0.06) and ef (0.04). Comparing posterior probability of hypothesis ab computed from the lattice (0.5) and from the WCN (0.25) we can conclude that probability distribution represented with WCN is very "blurred". This is caused by the fact that in this case the first word a fully determines the second word b and $P(ab) = P(b|a)P(a) \neq P(a)P(b)$.

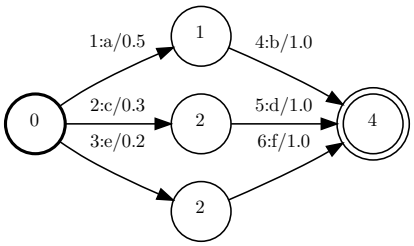

Fig. 1: Lattice example.

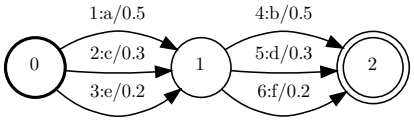

Fig. 2: Word confusion network example.

The presented comparison of lattices and WCNs is a motivation for a novel algorithm for semantic entity detection. The algorithm presented in the next section is able to represent accurately the posterior probability of multi-word substrings of ASR hypotheses. Because the input for an algorithm is a generic acyclic WFST, it can process both the word lattices and WCNs.

3. SEMANTIC ENTITY DETECTION

Preprocessing of input sentences with respect to lexical classes is a widely used approach in SLU systems. For example in the Semantic Tuple Classifier model [7], the lexical classes are replaced with class labels prior to training and decoding. The knowledge describing lexical classes is most often expressed as a list of lexical units pertaining to a given class. Such lists can be generated by hand or from domain databases. But the use of simple lists is limiting as there are many types of semantic entities which have more complicated structure (for example time or date entities). In this case only parts of the semantic entity are usually replaced with the corresponding lexical class, e.g. individual numbers but not the whole time entity. Such an approach is easy to implement and widely used. Due to its simplicity, it is limited to processing one-best ASR hypothesis or list of n-best hypotheses. This paper presents a new approach which allows to detect semantic entities described using a grammar (or finite state transducer) in a generic acyclic WFST (optimized lattice, WCN).

In current commercial dialog systems, it is common to represent the expert knowledge with probabilistic context free grammars (PCFGs) which are used both for speech recognition and understanding. The PCFG framework is also relatively simple to use – the developer of such grammar does not need to be a dialog system scientist to write a grammar with good coverage of target semantic entities. On the other hand, the probabilistic nature of PCFG is very rarely used because the expansion probabilities are very hard to determine using knowledge only – a large portion of recognition and understanding grammars are thus virtually deterministic context free grammars (CFG).

In this paper, we will use a set of CFGs to represent an expert knowledge about specific semantic entities. The use of expert knowledge is not limiting since the CFG for a given semantic entity can be transferred and shared between many dialog systems designed for different domains. In fact, the use of CFGs supports rapid devel-

```
$number = ($d | ten{10} | twenty{20}[$d]);
$d = (one {1} | two {2} | three {3});
$time = ten {10} past {p} three {3};
$year = last {2012} year;
```

Fig. 3: Example of SRGS grammars G_z for the following entities z: *number*, *time*, *year*. `$d` is an auxiliary rule. Grammar tags are in curly braces.

opment of a SLU module for a new dialog domain. The use of a knowledge-based CFGs does not imply that the method is not probabilistic – it allows to assign posterior probabilities to every semantic entity detected by CFGs.

Since it is common to represent the multiple ASR hypotheses as a WFST $\tilde{u}$, we first compile the knowledge base expressed as set of CFGs into a transducer which accepts a string of symbols representing exactly one semantic entity and transduces this string onto a sequence of semantic tags e_i. It is supposed that the first symbol in e_i indicates the type of semantic entity and the remaining symbols are its interpretation.

Each type of semantic entity z has a corresponding CFG G_z. Generally, the CFG is parsed using the pushdown automaton with unlimited stack depth. In the case where CFG is not recursive, the stack depth is limited and such an automaton can be converted into a FST where the input symbols correspond to CFG terminal symbols and output symbols are the so called *tags* assigned by the CFG. In this work, we use the standardized W3C speech recognition grammar specification (SRGS) [8] notation which allows to use tags inside the grammar specification (Fig. 3). The grammar G_z is converted to the unweighted finite state transducer T_z. The output symbols of T_z directly form the entity type and interpretation and allow the construction of "machine readable" objects (database entries, time objects) in a dialog manager. To represent the grammars G_z as a single transducer, we construct the union $Z = \bigoplus_z T_z$. The transducer Z can be subsequently optimized using generic WFST algorithms such as ϵ-removal, determinization and minimization.

The naive approach of generating the lattice $\tilde{e}$ from transducer composition $\tilde{u} \circ Z$ is infeasible because words not belonging to any semantic entity are not modelled by Z. The solution is to generate a factor automaton $F(\tilde{u})$ from the input lattice $\tilde{u}$ [9]. The factor automaton accepts all subpaths of the original lattice $\tilde{u}$. Then the composition $R = F(\tilde{u}) \circ Z$ generates the transducer where each path encodes a mapping from lattice subpath to a semantic entity.

The problem with such composition is that R contains many paths representing only partial matches of CFG – for example the utterance "twenty three" and grammar from Fig. 3 can lead to three different semantic entities of type number: *number:20*, *number:3*, *number:20:3*. It is easy to decide that only the last entity is meaningful in the context of the utterance.

Therefore we adopt the following *heuristics of maximum unambiguous coverage* – from the set of all possible semantic entities, we use the subset where (1) each transition in the lattice has assigned at most one semantic entity and (2) the number of transitions with assigned semantic entities is maximal.

Using the automaton R, it is possible to construct the set of all ambiguous semantic entities $\mathcal{F}$. Then the mentioned heuristics is applied to obtain the subset $\mathcal{F}^*$ containing only unambiguously assigned entities (Sec. 3.1).

In Sec. 3.2, we present an algorithm which takes $\mathcal{F}^*$ and time alignment of semantic entities to produce the WFST (lattice) $\tilde{e}$ where each path encodes one sequence e consisting of semantic entities. The weights of $\tilde{e}$ correspond to posterior probability distribution $P(E = e|W = \tilde{u})$.

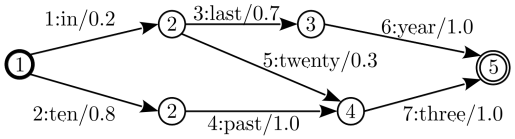

Fig. 4: Example of the transducer $\tilde{u}_T$, numbers assigned to states denote the state times, transitions labels correspond to the scheme: *input sym.*:*output sym.*/*weight*.

i	u^i	y^i	w_R	t_0^i	t_1^i
1	3, 6	**year:2012**	0.14	2	5
2	5, 7	**number:20:3**	0.06	2	5
3	5	number:20	0.06	2	4
4	2	number:10	0.80	1	2
5	7	number:3	0.86	4	5
6	2, 4, 7	**time:10:p:3**	0.80	1	5

Table 1: Example of the set $\mathcal{F}$ generated from $\tilde{u}_T$ with overlapping pairs (u^2, u^3), (u^2, u^5), (u^4, u^6) and (u^5, u^6). Highlighted semantic entities are members of $\mathcal{F}^*$.

3.1. Maximum unambiguous coverage

In order to keep the references to original states and corresponding time information stored in $\tilde{u}$ during the whole processing, the acceptor $\tilde{u}$ is mapped to a transducer $\tilde{u}_T$ by replacing input labels on each transition with globally unique identifier. We suppose that the acceptor $\tilde{u}$ is defined over *probabilistic semiring*. Then the factor transducer $F(\tilde{u}_T)$ is created and used in composition with the grammar transducer:

$$R = F(\tilde{u}_T) \circ Z \tag{1}$$

For each path $\pi^i \in R, i = 1, 2, \dots n$ with input symbols $i[\pi^i] = u^i$ and output symbols $o[\pi^i] = y^i$, the tuple $f^i = (u^i, y^i, w_R[u^i, y^i])$ is constructed. The weight $w_R[u^i, y^i]$ corresponds to a posterior probability of input sequence symbols u^i occurring in the lattice $\tilde{u}_T$.

Then the sequence $\mathcal{F} = \{f^i\}_{i=1}^n$ can be constructed. Each element of $\mathcal{F}$ represents one (partial) match of one of the grammars G_z and the lattice $\tilde{u}$ and each y^i is a semantic entity generated by G_z. The corresponding sequence u^i references the original transitions in $\tilde{u}$ which are matched. Since the hypotheses about the semantic entities contained in $\mathcal{F}$ are not necessary unambiguous, the integer linear programming (ILP) is used to obtain only the semantic entities matching the heuristics of maximum unambiguous coverage.

The ILP solver searches for the maximum of a criterion function $\mathbf{c}^\top \mathbf{x}$ given a vector of n binary variables $\mathbf{x} = [x_i]_{i=1}^n, x_i \in \{0, 1\}$ with respect to a set of constraints of the form $x_i + x_j \leq 1, i \neq j$. The constraints correspond to the first part (unambiguous) and the criterion to the second part (maximum coverage) of the heuristics. After the optimization is performed, the variable x_i is set to 1 if the path $\pi^i \in R$ matches the heuristics (y^i is a meaningful semantic entity).

The set of constraints contains the inequality $x_i + x_j \leq 1, i \neq j$ if and only if u^i and u^j is an *overlapping pair*. We say that string u^i overlaps u^j if (1) u^i is a substring of u^j or (2) there is a nonempty string u and strings a and b such that $u^i = au$ and $u^j = ub$. We use symmetrized relation – if u^i overlaps u^j or u^j overlaps u^i then u^i and u^j is an overlapping pair.

The optimization criterion is given by a vector $\mathbf{c}$ with components c_i:

$$c_i = |u^i|^2 \cdot w_R[u^i, y^i] \tag{2}$$

where $|u^i|$ denotes the number of symbols in u^i. This form of criterion ensures that the longer factors with higher posterior probabil-

ities are prioritized over the shorter and less probable ones. After solving the ILP optimization, the optimal subsequence $\mathcal{F}^*$ of sequence $\mathcal{F}$ is generated:

$$\mathcal{F}^* = \{f^i \in \mathcal{F} : x_i = 1\} \tag{3}$$

3.2. Construction of semantic entity lattice

To obtain the probability distribution $P(E|W)$, we need to generate the semantic entity lattice $\tilde{e}$ where each path corresponds to a sequence of semantic entities e and the associated weight to the probability $P(E = e|W = \tilde{u})$.

Apart from the sequence $\mathcal{F}^*$, the construction of $\tilde{e}$ requires state times t_0^i and t_1^i associated with starting and ending time of factor u^i in the original lattice $\tilde{u}_T$.

To describe the reconstruction algorithm, we need to define the forward ($\alpha[q]$) and backward ($\beta[q]$) probabilities in transducer T for a given state q [10]. The forward probability of an initial state and the backward probability of a final state is $\bar{1}$[1]. The forward and backward probabilities for the remaining states can be computed using:

$$\alpha[q] = \bigoplus_{t \in E: n[t]=q} \alpha[p[t]] \otimes w[t] \tag{4}$$

$$\beta[q] = \bigoplus_{t \in E: p[t]=q} w[t] \otimes \beta[n[t]] \tag{5}$$

where E is a set of all transitions in T. For a given transition $t \in E$, $p[t]$ is an origin state, $n[t]$ is a destination state and $w[t]$ as a weight assigned to t [11]. This notation can be extended from transitions to paths. The path π is defined as a sequence of transitions $t_1 t_2 \dots t_k$, therefore $p[\pi] = p[t_1]$, $n[\pi] = n[t_k]$, $w[\pi] = \otimes_{i=1}^{k} w[t_i]$.

The posterior probability of the factor u^i of the transducer $\tilde{u}_T$ can be expressed as:

$$P(u^i \in \tilde{u}_T) = \alpha[p[u^i]] \otimes w[u^i] \otimes \beta[p[u^i]] \tag{6}$$

Since the grammars G_z are deterministic, the transducer Z is unweighted and this probability distribution is not changed by the composition $F(\tilde{u}_T) \circ Z$; that is, we can express it as:

$$P(u^i \in \tilde{u}_T) = w_R[u^i, y^i] \tag{7}$$

The goal is to construct the lattice $\tilde{e}$ containing the semantic entity y^i with posterior probability $w_R[u^i, y^i]$. In the reconstruction algorithm, we treat the sequence y^i as a single symbol, e.g. *time:13:h*.

The algorithm starts with an empty transducer $\tilde{e}$. In the first step, the set of isolated transitions $t^i, i = 1, 2, \dots |\mathcal{F}^*|$ with corresponding symbols y^i is created. By the term isolated transition we mean that there is only one transition t^i leaving $p[t^i]$ and no transition leaving $n[t^i]$.

Then the set of so called *parallel states* is created. The state $p[t^i]$ has assigned a parallel state r^i with time $t[r^i] = t_0^i$ and $n[t^i]$ a parallel state b^i with time $t[b^i] = t_1^i$. The parallel states are connected with the origin and destination of transition t^i using ϵ-transitions from r^i to $p[t^i]$ and from $n[t^i]$ to b^i.

The parallel states are sorted according to assigned parallel state times. The result is the sequence $S = \{s_1, s_2, \dots s_{2n^*}\}$ where $t[s_i] \leq t[s_j]$ iff $i \leq j$. If two or more consecutive states have the same time, the b-states precede the r-states. The states s_1 and s_{2n^*} are marked as initial state and final state of $\tilde{e}$, respectively. Finally, the ϵ-transitions from s_i to s_{i+1} are added.

[1]We assume that the final weights of the used automata are equal to $\bar{1}$.

The last step is an assignment of weights so that $P(y^i \in \tilde{e}) = w_R[u^i, y^i]$. All transition weights are set to $\bar{1}$ except the transitions leaving r-states. The weights are determined using the forward probabilities $\alpha[q]$. The forward probabilities of states $p[t^i], i = 1, 2, \dots n^*$ are fixed to:

$$\alpha\left[p[t^i]\right] = w_R[u^i, y^i] \tag{8}$$

Then the forward probabilities are computed recursively for each state in $\tilde{e}$ starting from s_1. If the recursion reaches the state r^i, the weight of transition g^i from r^i to $p[t^i]$ is determined using

$$w[g^i] = \frac{\alpha\left[p[t^i]\right]}{\alpha[r^i]} \tag{9}$$

and the weight of transition h^i from r^i to the next state in the sequence S is equal to

$$w[h^i] = 1 - w[g^i] \tag{10}$$

The weight assignment assures that the $\oplus$-sum of weights of all paths in $\tilde{e}$ is $\bar{1}$ and also that the backward probability of each state is $\bar{1}$. Therefore the posterior probability of factor y^i occurring in $\tilde{e}$ is equal to

$$\begin{aligned} P(y^i \in \tilde{e}) &= \alpha\left[p[t^i]\right] \otimes w[t^i] \otimes \beta\left[n[t^i]\right] \\ &= \alpha\left[p[t^i]\right] = w_R[u^i, y^i] = P(u^i \in \tilde{u}_T) \end{aligned} \tag{11}$$

The acceptor $\tilde{e}$ represents the probability distribution $P(E = e|W = \tilde{u})$ over all possible paths $e \in \tilde{e}$. Since all transitions of $\tilde{e}$ except t^i are labeled with ϵ symbols, the ϵ-removal and determinization algorithms can be applied to obtain the optimized acceptor $\tilde{e}$. The optimization also sums overlapping semantic entity probabilities while preserving the probability distribution.

3.3. Processing time optimization

In this section we will shortly discuss the computational requirements of the proposed algorithm. The processing delay of the semantic entity detection is very important if such an algorithm is used in spoken dialog systems. There are two computationally demanding steps of the algorithm – the construction of the set $\mathcal{F}$ using the factor automaton and the selection of maximum unambiguous subset $\mathcal{F}^*$ using ILP.

The time required to construct $\mathcal{F}$ is mainly influenced by the size of the factor automaton R. To limit the size of $\mathcal{F}$, it is possible to prune the low-probability paths from R. In our experiments, we

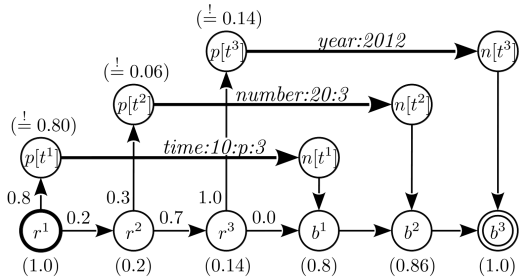

Fig. 5: Reconstruction of $\tilde{e}$ from the set $\mathcal{F}^*$, the bold transitions are *isolated transitions* which are created first. The values in parenthesis are forward probabilities $\alpha[q]$, three of them are fixed according to Eq. 8. Transition weights are computed using Eq. 9 and 10.

	raw	P-3	P-5	WCN
1-best Acc [%]	74.79	75.01	74.92	74.78
Oracle Acc [%]	91.26	84.62	87.60	90.24
# of states	68.15	7.68	12.37	8.40
# of transitions	555.05	9.84	24.03	34.40
Out-degree of states	8.14	1.28	1.94	4.10
WFST gen. time [ms]	–	1.05	1.25	17.72

Table 2: Comparison of various lattice representations. The values in last four rows are averages over the 1439 test sentences used in experiments. Note: the WCN generation is not optimized for speed.

type z	$\|A\|$	$\|B\|$	states	transitions	O
station	7516	3005	34405	5564	417
time	437	221	13375	2898	791
train_type	16	10	24	11	140

Table 3: Characteristics of semantic entity grammars compiled into optimized transducer T_z. $|A|$ and $|B|$ denote the size of an input alphabet and output alphabet, respectively, and O the number of occurrences of the particular semantic entity type in the used data set.

pruned the factor automaton R using the threshold $e^{-5} = 0.0067$ which ensures that there is no semantic entity hypothesis in $\mathcal{F}$ with posterior probability lower than this threshold.

After pruning the factor automaton, the set $\mathcal{F}$ can contain many hundreds of variables. The proposed algorithm uses the ILP solver to select the subset $\mathcal{F}^*$. Since solving the ILP is an NP-hard problem selecting the subset $\mathcal{F}^*$ has generally exponential complexity. On the opposite side we have found that only a small number of members of $\mathcal{F}$ has a higher posterior probability. Therefore we can limit the set $\mathcal{F}$ to N-best hypotheses prior to solving the ILP. In our experiments we used $N = 60$. In the experimental part we will shortly discuss the influence of this threshold on the semantic entity detection performance.

4. EXPERIMENTS

The experiments were performed on the Human-Human Train Timetable (HHTT) corpus [12]. The corpus contains inquiries and answers about train connections. The test set of HHTT corpus contains 1439 sentences of total length 45 minutes. The ASR recognition vocabulary contains 13,886 words and the accuracy of ASR on the test set is 74.8% with OOV rate 7.5%. We used the following representations of ASR hypothesis: the best hypothesis (1-best), the raw unoptimized lattice (raw), the optimized lattice generated using threshold $t_p = 3$ (P-3) and $t_p = 5$ (P-5) and word confusion network (WCN). To obtain the raw lattices, we used our in-house ASR decoder [13] which is able to compute directly the transition probabilities. In other words, the raw lattices are normalized, the probabilities of all hypotheses sums to one. For the characteristics of WFSTs see Tab. 2. It is obvious that the oracle accuracy of WCNs is similar to the original raw lattices and higher than the oracle accuracy of both variants of optimized lattices.

The grammars used in our experiments were described in [14]. We used three semantic entities: *station*, *time*, *train_type*. The interpretation of the entity *station* differentiates between departure, arrival and transit stations, entity *time* also contains date informations. The characteristics of corresponding transducers T_z (after

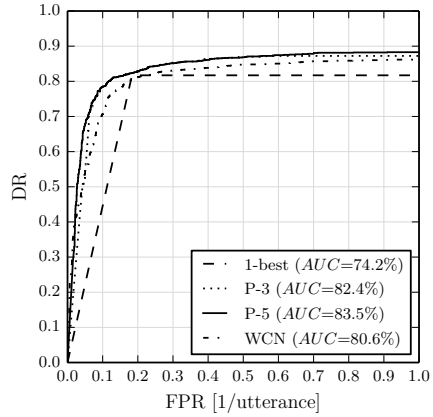

Fig. 6: ROC curves for different structures of ASR hypotheses.

semantic entity types	1-best	P-3	P-5	WCN
station	72.8	82.7	**83.5**	80.1
time	78.9	84.8	**85.8**	84.0
train_type	89.0	91.9	94.1	**94.8**

Table 4: *AUC* measures in percents for different semantic entities. Comparison of 1-best, optimized lattices and WCNs.

ϵ-removal, determinization and minimization) are shown in Tab. 3.

In our experiments, we did not evaluate the number of semantic entities covered by the expert-defined grammars. Instead we fixed the set of grammars and evaluated the influence of the ASR hypotheses representation on the detection performance. Since the HHTT corpus does not contain the manually tagged semantic entities, we first ran the algorithm on the manually annotated transcription to obtain the reference semantic entities.

Then the results of semantic entity detection from ASR hypotheses are evaluated against this reference. For each hypothesis, the posterior probability $P(y^i \in \tilde{e})$ is used as a score. We used the ROC curve to plot the dependency of detection rate (*DR*) on the false positive rate (*FPR*) which is normalized to express the number of false alarms per one utterance. The performance is measured using the computed area under the curve – *AUC*. The *AUC* measure represents the mean detection rate on the interval from 0 to 1 false alarm per utterance.

First, we evaluated the presented algorithm using the 1-best word string, optimized lattices P-3 and P-5 and WCNs as an input. The results are depicted in Fig. 6. It is obvious that the use of multiple hypotheses increases the mean detection rate measured by *AUC* from 74.2% for 1-best to 80.6% for WCNs and 83.5% for optimized lattices P-5. Table 4 compares the *AUC* measures for a given entity types. The only type for which the WCNs outperforms optimized lattices is *train_type*. This is caused by the fact that the train types are defined as a list of single words (such as *passenger*, *express*, etc.) and the WCNs correctly represent just the unigram posterior probability (see Sec. 2). The remaining semantic entities are sequences containing more than one word – stations are preceded by prepositions, times contain minutes and hours, etc. For these entities the optimized lattices approach outperforms the WCNs. To validate this hypothesis we split the reference data into two sets – one containing just semantic entities of one word length (len=1) and the rest containing longer entities (len>1). The ROC curves for

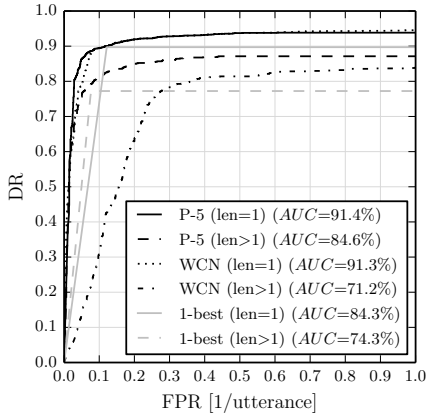

Fig. 7: ROC curves for semantic entities composed of a single word (len=1) and multi-word entities (len>1).

these experiments are shown in Fig. 7. For single-word semantic entities the curves for P-5 and WCN are almost overlapping. But for multi-word semantic entities the performance of WCN-based approach is poor (*AUC*=71.2%). The semantic entity detection which uses optimized lattices fully benefits from the use of factor automaton (*AUC*=84.6%). The factor automaton is able to exactly represent posterior probabilities of all factors regardless of the factor length.

5. CONCLUSIONS

We presented a fast algorithm for semantic entity detection. In the experiments, we limited the set of hypothesised semantic entities $\mathcal{F}$ to 60 best hypotheses. Using this threshold we are able to achieve *AUC*=83.5% by using the optimized lattices with pruning threshold $t_p = 5$ (P-5) and average processing time per one utterance 4.59 ms. By decreasing the threshold to $t_p = 3$ (P-3) the mean detection rate decreases to *AUC*=82.4% and processing time per one utterance decreases to 3.30 ms. Both these results are better than semantic entity detection from the 1-best hypothesis (*AUC*=74.2%) and than the approach based on word confusion networks (*AUC*=80.6%). By increasing the threshold to $N = 80$, the performance of WCN-based method changes only insignificantly to *AUC*=80.8% but the average processing time rises to 205 ms per utterance. For the optimized lattices the increase of N does not lead to any change in performance and the average processing increased to 5.22 ms.

We have also confirmed the theoretical problem (described in Sec. 2) that the WCNs are not able to model posterior probabilities of word strings longer than one word. The presented approach based on the use of factor automaton solves this issue. At the same time the computational requirements are in order of milliseconds which allows to use this method in real-time applications. The presented method based on expert defined CFGs allows rapid prototyping of a new dialog systems. The method allows to assign posterior probabilities to certain semantic entities. The algorithm can be used as a knowledge-based SLU itself or in combination with a statistical-based model such as [1].

6. ACKNOWLEDGEMENTS

This research was supported by the Technology Agency of the Czech Republic, project No. TE01020197.

7. REFERENCES

[1] Jan Švec, Luboš Šmídl, and Pavel Ircing, "Hierarchical Discriminative Model for Spoken Language Understanding," in *IEEE International Conference on Acoustics Speech and Signal Processing*, Vancouver, Canada, 2013, IEEE.

[2] Frédéric Béchet, Allen L Gorin, Jeremy H Wright, and Dilek Hakkani Tür, "Detecting and extracting named entities from spontaneous speech in a mixed-initiative spoken dialogue context: How May I Help You?," *Speech Communication*, vol. 42, no. 2, pp. 207–225, Feb. 2004.

[3] Christian Raymond, Frédéric Béchet, Renato De Mori, and Géraldine Damnati, "On the use of finite state transducers for semantic interpretation," *Speech Communication*, vol. 48, no. 3-4, pp. 288–304, Mar. 2006.

[4] Dogan Can and Murat Saraclar, "Lattice Indexing for Spoken Term Detection," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 8, pp. 2338–2347, 2011.

[5] Dilek Hakkani-Tür, Frédéric Béchet, Giuseppe Riccardi, and Gokhan Tur, "Beyond ASR 1-best: Using word confusion networks in spoken language understanding," *Computer Speech & Language*, vol. 20, no. 4, pp. 495–514, Oct. 2006.

[6] Natthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young, "Discriminative Spoken Language Understanding Using Word Confusion Networks," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*, 2012, pp. 176–181.

[7] 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.

[8] A. Hunt and S. McGlashan, "Speech recognition grammar specification version 1.0," W3C Recommendation, Mar. 2004.

[9] Mehryar Mohri, Pedro Moreno, and Eugene Weinstein, "Factor automata of automata and applications," *Implementation and Application of Automata*, vol. 4783, pp. 168–179, 2007.

[10] Mehryar Mohri, "Semiring frameworks and algorithms for shortest-distance problems," *Journal of Automata, Languages and Combinatorics*, vol. 7, pp. 321–350, 2002.

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

[13] 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.

[14] Tomáš Valenta, Jan Švec, and Luboš Šmídl, "Spoken Dialogue System Design in 3 Weeks," *Text, Speech and Dialogue*, vol. 7499, pp. 624–631, 2012.