A ONE-STAGE DECODER FOR INTERPRETATION OF NATURAL SPEECH

Matthias Thomae, Tibor Fabian, Robert Lieb, Günther Ruske

Institute for Human-Machine Communication
Technische Universität München, Germany
{tho,fab,lie,rus}@mmk.ei.tum.de

ABSTRACT

Current speech understanding systems are typically designed as multi-stage systems, although this theoretically gives rise to errors due to early decisions. We present a framework that offers the chance of reducing these errors by an integrated system which directly computes a semantic tree representation from the input speech signal through a token passing based one-stage decoder, called ODINS. In order to limit the complexity of ODINS, we represent all a-priori knowledge consistently by a generalized uniform knowledge model based on a hierarchy of probabilistic transition networks, which also can be n-grams. Our framework includes a method to evaluate the system output using an edit distance based tree matching algorithm. First experiments quantify and confirm the theoretical advantage of the one-stage strategy over a corresponding two-stage approach.

1. INTRODUCTION

The incorporation of higher-level linguistic knowledge into speech recognition systems is receiving increasing attention. Popular application fields like information access tasks require some representation of the meaning of the user's utterances rather than the pure orthographic transcription. Moreover, it has become clear that new information sources must be utilized to further enhance speech recognition performance, especially when the speech signal is disturbed. Typically, the higher linguistic knowledge is applied sequentially, after a speech recognition module has computed a word lattice representation of a spoken utterance. This strategy is often preferred over a one-stage approach for computational or practical reasons, or because of difficulties to integrate the linguistic models into the first stage. Some examples of such multi-stage systems are EVAR, LIMSI ARISE, MIT JUPITER/VOYAGER, SRI ATIS, SUNDIAL and TRAINS/TRIPS.

However, recent publications in the field of speech recognition report that one-stage decoders can be as efficient as or even more efficient than multi-stage systems, even for large vocabulary tasks [1, 2]. They also indicate that the theoretical advantage of a one-stage decoding strategy, based on applying all available knowledge sources as early as possible and simultaneously, is of practical relevance regarding its performance, both in the sense of runtime and recognition accuracy.

In this paper we are presenting our work on a one-stage decoding framework for speech interpretation, aiming at limited-domain applications with medium-sized vocabularies. It is based on representing all available knowledge sources, from the phoneme models to the semantic units, by a generalized, unified hierarchical network. The concept of *generalization* offers flexibility and extensibility regarding the integration of various sources of knowledge into the speech interpreter, without having to change the decoder itself. The concept of a *unified knowledge representation* also aims at rendering decoder modifications unnecessary, but moreover at limiting decoder complexity. The concept of a *hierarchical representation* aims at modularization, i.e. the possibility to develop individual knowledge sources independently of each other. The idea of a generalized, unified, hierarchical approach is comparable and partly based on previous work [3, 4]. However we introduce several novel aspects that enhance the already published ideas.

In Section 2 we discuss the underlying assumptions and theory that lead to the generalized uniform modeling approach. With the aid of an example, we then demonstrate the structure of the generalized transition network hierarchy and discuss its construction for a speech interpretation task. Based on a data collection for a spoken dialogue scenario in the domain of airport information services, we built a uniform, hierarchical network consisting of local models for phonemes, words, word classes and semantic concepts, and a global sentence model in the form of an n-gram network. Section 3 presents our current one-stage decoder implementation for such models. It follows the well-known token passing paradigm [3] to perform a time-synchronous Viterbi search. Our implementation aims at a compact memory representation by a shared representation of the network topology.

Sophisticated linguistic knowledge representations utilize the expressive power of context free grammars (CFGs) or even augmented context free grammars such as unification grammars [5]. Although a hierarchical network can generally represent arbitrary CFGs, our current decoder implementation doesn't permit the use of recursive CFGs, because this restriction simplifies the decoding algorithm and because the linguistic knowledge we used so far is rather simple. However, there are efficient decoder designs for CFGs without left recursion [6, 7]. As left recursion can be removed from CFGs by conversion to Greibach normal form [5, 8], Viterbi decoding of arbitrary CFGs is generally possible.

The token passing algorithm can readily be extended for the generation of multiple alternative hypotheses [3]. For speech recognition tasks, these are required to generate word lattices, that can be used for a number of purposes, e.g. as interface between speech recognizer and linguistic parser or for the computation of confidence measures. We also extended our decoder to output multiple alternatives, but generalized the word lattice scheme so that hierarchical lattices, composed of a root lattice and several sublattices, can be generated (see Section 3.3). In comparison to other

This work was funded partly by the German Research Council (DFG) project Ru 301/6-1 and also by the NADIA research project from the Bayerische Motorenwerke (BMW) group.

From: NLP-KE 03, IEEE Press

concept-based lattices [9] a hierarchical lattice can have more than one level of sub-lattices, and, following our hierarchical network representation, sub-lattices may be shared directly or indirectly.

We applied hierarchical lattices in a novel experiment quantifying the accuracy gain of a one-stage speech interpreter over a corresponding two-stage approach. In order to measure only the effects resulting from the simultaneous application of all knowledge sources, we modified the token passing algorithm so that it can be used for the decoding of hierarchical lattices. The two-stage system was then composed of a speech recognizer that generates hierarchical lattices consisting of word and word class labels, and a linguistic model stage that conducts the search through the hierarchical language model constrained by the hierarchical lattice. The constrained token passing algorithm will be discussed briefly in Section 3.4. We also apply this algorithm to automatically generate hierarchical annotations of spoken utterances that were previously transcribed orthographically.

The output of the one-stage interpreter is an ordered, labeled tree of words and semantic units, called *semantic tree*. In Section 4 we define an evaluation metric that especially takes the structure of the semantic trees into consideration, and discuss why it might be more adequate than usually used metrics.

In Section 5 we present some evaluation results, especially for a comparison of one-stage and two-stage systems.

2. GENERALIZED NETWORK HIERARCHY

The basic assumption that leads to a generalized hierarchical transition network model is that of a sequential correspondence between the input signal $\mathbf{I}$ and the output signal $\mathbf{O}$ [10]. Given that the input and output signals consist of sequences of basic units $\mathbf{I} = i_1, i_2, \ldots, i_{N_i}$ and $\mathbf{O} = o_1, o_2, \ldots, o_{N_o}$, a sequential correspondence between $\mathbf{I}$ and $\mathbf{O}$ means that $\mathbf{I}$ can be segmented into consecutive sub-sequences $\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_{N_o}$, so that each sub-sequence $\mathbf{i}_k$ directly corresponds to an output unit o_k ($k = 1 \ldots N_o$, $N_o \leq N_i$). In a speech interpretation system $\mathbf{I}$ are the acoustic observations and $\mathbf{O}$ is a sequence of semantic concepts. However, the concept likelihoods are not directly estimated from the acoustic observations, but intermediate levels of representations are introduced, such as phonemes or words.

2.1. Generalization

In order to generalize the (intermediate) levels of representation to an arbitrary number N, we assume that the levels pairwise correspond sequentially. A level with index m consists of a sequence of N^m basic units $\mathbf{L}^m = l_1^m, l_2^m, \ldots, l_{N^m}^m$ which can be segmented into consecutive sub-sequences $\mathbf{l}_1^m, \mathbf{l}_2^m, \ldots, \mathbf{l}_{N^{m+1}}^m$ so that each sub-sequence $\mathbf{l}_n^m$ directly corresponds to a unit l_n^{m+1} on the next higher level. The level index m ($m = 1...N, N \geq 2$) increases from the lowest level ($m = 1$) towards the highest level ($m = N$). The units and their correspondences form a general hierarchical (tree) structure. Given this structure and assuming that each level only depends on the next higher one, the goal of finding the optimum output sequences $\mathbf{L}^2, \mathbf{L}^3, \ldots, \mathbf{L}^N$ can be expressed in a probabilistic way by the maximum a-posteriori formulation:

$$\underset{\mathbf{L}^2,\mathbf{L}^3,\ldots,\mathbf{L}^N}{\arg\max} \Big(\prod_{m=1}^{N-1} P(\mathbf{L}^m|\mathbf{L}^{m+1})\Big) P(\mathbf{L}^N) \qquad (1)$$

With the sequential correspondence assumption we can further decompose the likelihood of the correspondence between a pair of unit sequences $P(\mathbf{L}^m|\mathbf{L}^{m+1})$ into the likelihoods of the sub-sequence correspondences $P(\mathbf{l}_n^m|l_n^{m+1})$:

$$\underset{\mathbf{L}^2,\mathbf{L}^3,\ldots,\mathbf{L}^N}{\arg\max} \Big(\prod_{m=1}^{N-1} \prod_{n=1}^{N^{m+1}} P(\mathbf{l}_n^m|l_n^{m+1})\Big) P(\mathbf{L}^N) \qquad (2)$$

As mentioned in the introduction, the generalization to an arbitrary number of hierarchy levels offers flexibility and extensibility without the need for decoder modifications. This is especially helpful if the incorporation of novel knowledge sources into a one-stage decoder is to be examined.

We further extended the flexibility by allowing each hierarchy level to be composed of an arbitrary number of sub-levels. Thus, for example a semantic concept may be composed of a sequence of sub-concepts, which in turn may be composed of further sub-concepts, and so on. While hierarchy levels and their sub-levels do not differ with respect to Expression (2), they do regarding their properties. By definition, all sub-levels of a hierarchy level share the same properties, whereas different hierarchy levels may have different properties. These properties can be search parameters or structural restrictions. For example, we restrict word classes to contain exactly one word, or language model factors to be only applied to syntactic and semantic levels.

Another measure to raise the flexibility of the hierarchy is to allow the skipping of sub-levels or even entire levels. Skipping of sub-levels can be useful at the semantic level, for example, so that a concept may appear both at the surface and as part of a higher conceptual category, such as *AFlightNumber* in Figure 1. Examples for skipping an entire level can be seen in Figure 1, where only some of the words such as digits or airline codes belong to word classes, whereas others such as *wann* are referred to directly from concepts. Note that entire level skips can prevent the separate application of level properties, especially search parameters. Generally however, level skips add to the flexibility of the approach and result in a more compact hierarchy representation because otherwise trivial models containing only one unit would have to be added. Regarding the maximum a-posteriori formulation both representations are equivalent, as the likelihood of a skipped trivial model is one.

2.2. Uniform network representation

We represent all knowledge models (e.g. phonemes, words, word classes and concepts) uniformly as a hierarchy of weighted transition networks. As in [3] networks consist of three types of nodes: Non-terminal or *sub-network nodes*, epsilon or *null nodes* and *terminal nodes*. A sub-network node refers to another network, called the *sub-network* of the node. Regarding Expression (2), the sub-network of a node l_n^{m+1} is the model for $P(\mathbf{l}_n^m|l_n^{m+1})$, the likelihood for a sequence of corresponding basic units $\mathbf{l}_n^m$. It is computed by the product of the transition probabilities along the path corresponding to $\mathbf{l}_n^m$. We assume that each network has exactly one entry and one exit node, an arbitrary number of null nodes and at least one sub-network or terminal node. For a speech processing task, terminal nodes are the points where time-frames of the speech signal are consumed. When representing Hidden-Markov Models (HMMs) within this framework, a terminal node corresponds to an HMM state. Viewing the networks themselves as nodes and the references from the sub-network nodes to their sub-networks as edges yields the *super-network*. The super-network describes the dependencies between the networks and thus represents the structure of the network hierarchy. Each network has a

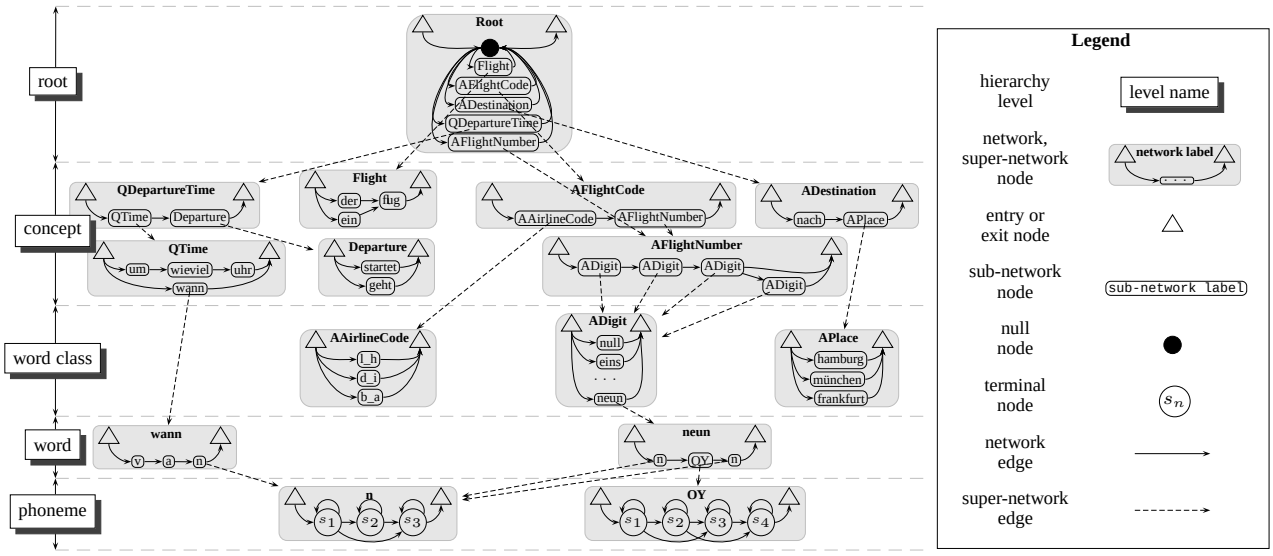

Fig. 1. *Simple example for a hierarchical transition network that for example accepts the utterance 'wann startet der flug l_h drei sieben neun eins nach hamburg' (Literally: when starts flight l_h three seven nine one to hamburg).*

label and belongs to a certain *hierarchy level*, for example to a phoneme, to a word, or to a conceptual category. Figure 1 shows a simple example of such a hierarchical transition network for a speech interpretation task. For clarity of illustration, no transition weights are shown and only a small part of the sample network is displayed so that not all paths through the super-network end at a terminal node. All complete paths through the hierarchy start at the entry node of the global *root network* and end at its exit node. If a path arrives at a sub-network node, it continues at the start node of its sub-network. Similarly, paths arriving at an exit node continue at the sub-network node they once descended from.

2.3. Network construction

The network hierarchy can be constructed in a number of ways. The different approaches can be characterized as being rule-based or data-driven, or both. A manually built rule or grammar can be probabilized by estimating occurrence frequencies on a corpus. Moreover, rule-based and data-driven approaches can be combined in different parts of the grammar or network hierarchy, so that the optimum (best performing, least time consuming) solution for the task at hand can be chosen. The only requirement for a model is that it is representable as a probabilistic (possibly hierarchical) transition network. Among the commonly used ones are backing-off n-grams, (discrete) HMMs, regular rules and grammars and context-free grammars (with the limitations discussed in Section 1). As the acoustic-phonetic models and the pronunciation lexicon follow standard approaches from speech recognition, we here focus on the speech interpretation part, i.e. the semantic modeling (e.g. the word class, concept and root levels in Figure 1). We also refer to this part as the *hierarchical language model*. A data-driven approach to build such a language model requires a hierarchically annotated speech data collection (except for the case that the hierarchy is generated automatically). The annotation is based on the word level and extended by hierarchically grouping sequences of basic units into equivalent semantic categories. The sequential correspondence assumption implies that the semantic annotation forms a strict tree structure. Therefore, no overlapping annotations are allowed, so that the annotation tree has no crossing edges. The node labels of the annotation tree correspond to the network labels. An example for an annotation tree can be seen in Figure 2 (top).

We used a semi-automatic, iterative procedure to hierarchically annotate a speech corpus for an airport information system dialogue scenario. Based on a small manually annotated part of the corpus an initial hierarchical language model was built. With the aid of the constrained token passing algorithm that will be briefly introduced in Section 3.4, the hierarchical annotation of new utterances was generated from the word sequence automatically. Errors were then corrected by hand, and a new intermediate hierarchical language model was built on the extended annotations. This process was repeated iteratively for chunks of the corpus, as the tagging accuracy tends to improve with the incorporation of new phenomena into the intermediate language models. From the hierarchically annotated speech corpus a collection of sub-sequences can be extracted for each semantic category. They form the basis for the creation of the individual networks.

For the global network at the root of the hierarchy we employed a backing-off n-gram model. For the local networks data-driven and rule-based approaches were combined. The concept sub-networks are either created from manually formulated regular expressions, or by directly compiling the union of the concept's sub-sequences from the hierarchically annotated training set into a network. For both methods, the AT&T Finite-State Machine (FSM) toolkit [11] is utilized to compact the networks for an efficient decoding process. The transitions within the local concept networks are weighted according to their frequency of occurrence in the training set. In order to smooth these frequencies, we currently use a simple additive smoothing scheme. Word classes were selected manually for categories relevant to the task, like place names, airline and area codes, plane types and digits. Word classes

are also an essential requirement to scale the system's knowledge about target information, e.g. to be able to add place names that did not occur in the training data. Consequently, and also because target information is unevenly distributed in the data collection, words were uniformly weighted within their classes.

3. ONE-STAGE DECODING

Our *One-stage Decoder for Interpretation of Natural Speech* (ODINS) is based on the token passing paradigm [3] to perform a time-synchronous Viterbi search by token passing within a transition network structure. The token passing approach especially considers the part of the search process where no 'time' is consumed, i.e. outside the HMM states.

3.1. Basic token passing algorithm

The set of tokens that are active at a certain point during the search can be seen as the heads of search paths. The tails of the search paths are represented by back-tracking information which is typically realized as a linked list of so-called *back-tracking records*. The process of passing the tokens along network nodes is called *token propagation*. Tokens essentially consist of two pieces of information, namely a reference to their current back-tracking record, as well as an *accumulated score*. The accumulated score consists of the sum of terminal node (emission) and edge (transition) scores collected along the search path. In a single propagation step, a token is copied from it's current node to all successor nodes, updating the accumulated score with the respective edge's transition score. At the terminal nodes, the (log) acoustic emission score is added as well. At the end of the utterance, the best path through the search space is represented by that token collected from the exit node of the root network.

The search progresses with the propagation of tokens through the network hierarchy. This process can be divided into two main phases: A time-frame consuming phase, where all tokens at terminal nodes are propagated once, and a phase between frames, where tokens are generally propagated several times. The latter can be subdivided into downward propagation, where networks are processed in the topological order of the super-network, and upward propagation, where networks are processed in reverse topological order.

3.2. Implementation

Our version of the token passing algorithm differs from other formulations [3, 4] in several aspects. A fundamental difference is that we share the topology of the sub-network models among all referring sub-network nodes, such as for the words *wann* and *neun* in Figure 1 which share the phoneme *n*. Hence, multiple tokens simultaneously reside at a network node in general and can therefore be propagated together. Another basic difference is that our algorithm is formulated not in a recursive manner but iteratively, so that each network is only processed once during a propagation phase. To achieve this, a topological sort of the super-network is carried out; this implies that the network hierarchy is non-recursive. As mentioned in the introduction, we opted for this solution because decoding efficiency was more important for the task at hand than the use of recursive rules. A further reduction of the processing effort is achieved by topologically sorting networks themselves if possible, i.e. if the network is acyclic. In this case each node of a network needs only one processing step during a propagation phase.

Generally, tokens that traverse different paths through the search network may meet at the same node. An important property of the Viterbi search is that these paths may be *recombined*, i.e. only the best path survives, all others are discarded. Due to the sharing of the sub-network models paths may stem from different parent networks, thus the corresponding tokens must not be recombined. The criterion for recombination is what we call the *super-network history* of a token. Simultaneously it is utilized to decide which parent sub-network node a token must return to when it reaches the exit node of a sub-network. The super-network history is defined as that part of a token's path through the super-network which only considers the last downward propagation step from each network, starting with the last visit of the root network. More precisely, the last downward traversed sub-network *nodes* have to be considered in order to discriminate multiple (direct or indirect) references between two networks, such as between *AFlightNumber* and *ADigit* in Figure 1. We realize the super-network history as an unordered tree whose nodes correspond to the sub-network nodes of the hierarchy. Hence our tokens have an additional piece of information in the shape of a reference to a node of the super-network history tree. Only those tokens at a network node that refer to the same super-network history tree node can be recombined.

3.3. Hierarchical lattice generation

The basic token passing algorithm can be extended in order to produce multiple alternative hypotheses by storing not only the best, but the n-best tokens at each word boundary [3]. Hence the back-tracking information is no longer a linked list but an (acyclic) network. We extended our decoder in a similar way, but generalized the approach by storing n-best tokens at different levels in the network hierarchy, so that generalized *hierarchical lattices* can be produced. Similar to the network hierarchy they consist of a global lattice and several *sub-lattices*. Lattices that contain semantic information in addition to words have been proposed before (e.g. [9]), but our hierarchical lattices have two special properties: Firstly, they may consists of an arbitrary number of levels, and secondly, we can straightforwardly produce hierarchical lattices whose sub-lattices are directly or indirectly shared, thus yielding a more compact representation.

3.4. Constrained token passing

We also modified the token passing search so that it can be constrained to the paths of a (hierarchical) lattice. In contrast to the unconstrained token passing search only those paths are allowed which contain valid paths through the constraining lattice. This can be achieved by augmenting the tokens with a reference to the previously traversed terminal node of the lattice (terminal nodes are e.g. words). When a token reaches a terminal node, the token is only kept if the node is a possible successor of the previously traversed node. Moreover, due to the possibly hierarchical nature of the lattice, a token is discarded if its super-network history doesn't contain the possible *super-lattice histories* of the terminal lattice node. This measure ensures that all levels of the hierarchical lattice, not only the terminal symbols, constrain the possible paths. We used the constrained token passing search for two tasks: Firstly, to simulate a two-stage decoder that corresponds as much

as possible to its one-stage counterpart (see Section 5.1). Secondly, to automatically generate hierarchical annotations of spoken utterances that were previously transcribed (see Section 2.3).

4. EVALUATION METRICS

The performance of a speech recognition system is typically evaluated by automatically comparing reference transcriptions of utterances of a test corpus against the system output. This is usually carried out by computing the best match between the reference and hypothesis word sequences through the minimization of the edit distance. The edit distance is given by the sum of the costs for the three basic edit operations (insertion, deletion and substitution) which are required to transform the hypothesis into the reference sequence. We set these costs according to the widely used NIST scoring software [12] to 4 for substitutions and 3 for insertions and deletions. The minimization of the edit distance is usually carried out by dynamic programming. The *word accuracy* Acc_w is then computed from the counts of correct C_w, substituted S_w, inserted I_w and deleted D_w words, where $N_w = C_w + S_w + D_w$ is the number of words in the reference [13]:

$$Acc_w = \frac{N_w - S_w - I_w - D_w}{N_w} = \frac{C_w - I_w}{C_w + S_w + D_w} \quad (3)$$

Speech interpreters can be evaluated by a similar evaluation metric called *concept accuracy* Acc_c, where semantic concepts in the shape of slot–value pairs [14] are the basic units instead of words:

$$Acc_c = \frac{C_c - I_c}{C_c + S_c + D_c} \quad (4)$$

Both the slots and the values of concepts must match to be counted as correct C_c, a substitution S_c is given if only the slot matches, if it doesn't a deletion D_c or insertion I_c is counted. The output trees of ODINS can be converted accordingly, so that the concept accuracy can be computed. Each leaf node is taken as a value for the slot given by the concatenation of its ancestor nodes.

However, since trees can be seen as a generalization of sequences we propose an evaluation metric more adequate for tree structures which directly computes the best match between trees. This task is addressed by a class of algorithms that minimize the edit distance between labeled, ordered trees, and include sequence matching as a special case [15]. The edit distance between two trees is now determined by the edit operations necessary to transform one tree into another. The substitution operation changes a tree node label; a deletion removes a node and makes its children the children of its parent; inserting a node makes it the child of a certain node, and a subsequence of this node's children become the children of the inserted node. We set the costs for insertions and deletions to 3 as for the sequence matching case above. However, as we wish to avoid substitutions between tree nodes of different hierarchy levels, we only use a substitution cost of 4 if the corresponding nodes have the same level type. Otherwise, the substitution cost is set to ∞.

From the tree matching results the number of correct C_n, substituted S_n, inserted I_n and deleted D_n tree nodes can be computed. Consequently we define the *tree node accuracy* Acc_n analogously to Equation (3) as:

$$Acc_n = \frac{C_n - I_n}{C_n + S_n + D_n} \quad (5)$$

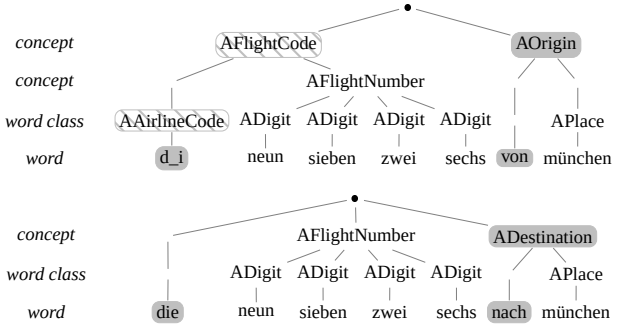

Fig. 2. *Tree matching example of reference (top) and hypothesis (bottom) trees.*

Figure 2 depicts an example for the alignment of reference and hypothesis trees by a tree matching algorithm. The minimum cost edit operations required to convert the reference into the hypothesis tree consist of deleting nodes *AFlightCode* and *AAirlineCode* and substituting *d_i* with *die*, *von* with *nach* and *AOrigin* with *ADestination*. All the other tree nodes are rated as correct, so that the tree node accuracy is $Acc_n = \frac{11-0}{11+3+2} = 68.75\%$.

In order to compute the concept accuracy, both trees are converted to sequences of 7 slot–value pairs as described above (concatenation denoted by a dot), e.g. (*AFlightCode.AAirlineCode*, *d_i*). As none of the concept's slots match, 7 deletions and 7 insertions are counted so that $Acc_c = \frac{0-7}{0+0+7} = -100.0\%$. Thus the concept accuracy doesn't rate the correctly recognized subtrees rooted at *AFlightNumber* and *APlace* as correct, in contrast to the tree node accuracy.

We argue that the tree node accuracy can be a more adequate evaluation metric than the concept accuracy, because it takes the *structural correspondence* into account. In conjunction with a confidence measure that would ideally rate *die* and *nach* as incorrectly recognized, a spoken dialogue system might ask the user for the missing airline code and whether *münchen* is the origin or the destination. Thus, partial information at lower hierarchy levels can also be made use of and should therefore be rated accordingly. Yet it can be argued if errors should be weighted in a non-uniform way, e.g. depending on the hierarchy level.

5. FIRST EVALUATION RESULTS

The one-stage decoding framework was first applied to a sample scenario of an airport information system. For this domain, spontaneous speech utterances were collected by simulating the dialogue system through a wizard-of-oz setup. A subset of the collected speech data of 1446 user utterances from 23 different speakers was annotated with semantic trees. From this material, the utterances of 17 speakers were used for training and 3 speakers were used for evaluation and cross-validation, respectively. The lexicon consists of 574 words, 138 of them are assigned to 11 different word classes. The tree annotation contains 47 different semantic concepts.

The acoustic model part consists of speaker-independent tied intra-word triphone HMMs with about 25k Gaussian mixture components in total. It was initially trained on a larger spontaneous speech corpus from the German Verbmobil project [16] and then adapted to 9 of the training speakers of the sample scenario. The

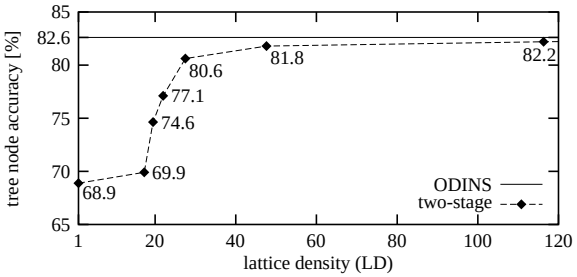

Fig. 3. *Tree node accuracy of ODINS vs. two-stage system.*

preprocessing stage computes 12 MFCC components, which together with energy, delta and acceleration coefficients yield 39-dimensional feature vectors. The hierarchical language model utilizes a backing-off bigram root network and was constructed as described in Section 2.3. The concept hierarchy level consists of 3 sub-levels. For this setup, the tree node accuracy as defined in Section 4 is 82.6% on the evaluation data.

5.1. One-stage vs. two-stage system

Additionally, we performed an experiment to estimate the superiority of ODINS over a classical two-stage approach. To make this experiment as fair as possible, the two systems should not only rely on the same training data, but also use the same modeling approach and the same type of decoding algorithm.

In order to achieve this, the network hierarchy was split into two parts: For the first stage we estimated a word class based bigram language model from the same training data as used for the hierarchical language model of ODINS, but now ignoring the conceptual annotations. For the second stage we used the same concept based hierarchical language model as for the one-stage setup. The interface between the two stages is a hierarchical lattice consisting of words and word classes. On this lattice, the second stage is performed with the constrained token passing search.

For both systems, the tree node accuracy is measured on the evaluation data. Figure 3 shows the results for the two-stage system, depending on the lattice density (LD), compared to the tree node accuracy of the one-stage system of 82.6%. The LD is defined as the total number of hypothesized units (words and word classes) divided by the number of units of the best path.

6. CONCLUSION

The experimental results show a clear advantage of our *One-stage Decoder for Interpretation of Natural Speech* (ODINS) at low LDs. For the case that the lattice consists of the best path only (LD=1), the one-stage system performs about 14% (absolute) better than the two-stage system. The two-stage system only approximately reaches the performance of ODINS when the lattice contains many alternatives.

The results quantify the theoretical advantage of the one-stage approach which applies all available knowledge as early as possible and thus avoids errors that occur due to missing semantic knowledge in the first stage of a two-stage system. In order to conduct the experiment the decoder was extended to produce generalized hierarchical lattices that may consist of an arbitrary number of levels and have shared sub-lattices, similar to the hierarchical search network. We also modified the token passing search to operate on a search network constrained by the paths of a hierarchical lattice, and reused this algorithm to support the hierarchical annotation procedure. Furthermore, we proposed and used a novel evaluation metric that is especially tailored for tree structured representations.

7. REFERENCES

[1] X. Aubert, "One Pass Cross Word Decoding for Large Vocabularies based on a Lexical Tree Search Organization," in *Proc. Eurospeech*, Budapest, Hungary, 1999, pp. 1559–1562.

[2] H. Soltau, F. Metze, C. Fügen, and A. Waibel, "A one-pass decoder based on polymorphic linguistic context assignment," in *Proc. ASRU*, Trento, Italy, 2001.

[3] S.J. Young, N.H. Russell, and J.H.S. Thornton, "Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems," Tech. Rep., CUED, July 1989.

[4] B. Jelinek, F. Zheng, N. Parihar, J. Hamaker, and J. Picone, "Generalized Hierarchical Search in the ISIP ASR System," in *Asilomar 2001*, 2001, vol. 2, pp. 1553–1556.

[5] R. C. Moore, "Using Natural Language Knowledge Sources in Speech Recognition," in *Proceedings of the NATO Advanced Study Institute (ASI)*. NATO, 1998.

[6] Nuance Communications, Menlo Park, California, *Nuance Speech Recognition System, Version 7.0, Grammar Developer's Guide*, 2001.

[7] X. Huang, A. Acero, F. Alleva, M.-Y. Hwang, L. Jiang, and M. Mahajan, "Microsoft Windows Highly Intelligent Speech Recognizer: Whisper," in *Proc. ICASSP*, Detroit, Michigan, 1995, pp. 93–96.

[8] J. E. Hopcroft and J. D. Ullman, Eds., *Introduction to Automata Theory, Languages and Computation*, Addison Wesley, 1979.

[9] K. Hacioglu and W. Ward, "A Concept Graph Based Confidence Measure," in *Proc. ICASSP*, Orlando, Florida, May 2002, pp. 225–228.

[10] R. Pieraccini, E. Levin, and E. Vidal, "Learning how to Understand Language," in *Proc. Eurospeech*, Berlin, Germany, 1993, pp. 1407–1412.

[11] M. Mohri, F. C. N. Pereira, and M. Riley, "A Rational Design for a Weighted Finite-State Transducer Library," in *Workshop on Implementing Automata*, 1997, pp. 144–158.

[12] William M. Fisher and Jonathan G. Fiscus, "Better Alignment Procedures for Speech Recognition Evaluation," in *Proc. ICASSP*, Minneapolis, Minnesota, 1993, pp. 59–62.

[13] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.2)*, CUED, 2002.

[14] M. Boros, W. Eckert, F. Gallwitz, G. Görz, G. Hanrieder, and H. Niemann, "Towards Understanding Spontaneous Speech: Word Accuracy vs. Concept Accuracy," in *Proc. ICSLP*, Philadelphia, PA, 1996, vol. 2, pp. 1009–1012.

[15] D. Shasha and K. Zhang, "Approximate Tree Pattern Matching," in *Pattern Matching Algorithms*, A. Apostolico and Z. Galil, Eds., chapter 14. Oxford University Press, 1997.

[16] W. Wahlster, Ed., *Verbmobil: Foundations of speech-to-speech translations*, Springer, Berlin, Germany, 2000.