Hidden Markov model–based Supertagging in a user–initiative dialogue system

Jens Bäcker and Karin Harbusch
University of Koblenz–Landau, Computer Science Department
Universitätsstr. 1, D–56070 Koblenz, Germany
E–mail: {jbaecker|harbusch}@informatik.uni-koblenz.de

Abstract

In this paper we outline the advantages of deploying a shallow parser based on Supertagging in an automatic dialogue system in a call center that basically leaves the initiative with the user as far as (s)he wants (in the literature called *user–initiative* or *adaptive* in contrast to *system–initiative* dialogue systems). The Supertagger relies on a *Hidden Markov model* and is trained with German input texts. The entire design of a Hidden Markov–based Supertagger with trigrams builds the central issue of this paper. The evaluation of our German Supertagger lags behind the English one. Some of the reasons will be addressed later on. Nevertheless shallow parsing with the Supertags increases the accuracy compared to a basic version of KoHDaS that only relies on recurrent plausibility networks.

1. Introduction

Wizard–of–Oz experiments show that users of automatic dialogue systems would preferentially take the initiative in many dialogues instead of being asked a long list of tiny little questions by the system (cf. (Boje *et al.*, 1999)). Empirical evaluations demonstrate that adaptation to the user's dialogue preference leads to significantly higher user satisfaction and task success (cf. (Strachan *et al.*, 1997) or (Litman, Pan and Walker, 1998)). In contrast to these results, it can also be observed that in such *user–initiated dialogue systems* the user is sometimes left without a clear understanding of his/her options at a given point in the dialogue. This can cause frustration or even breakdowns of the communication. Consequently, an *adaptive system* which reacts to the user's preferred mode, i.e. is able to ask explicit questions when the user doesn't take the initiative and to react to user–provided complex turns adequately as well at any particular state of the dialogue, serves as a user–friendly dialogue system.

The criticised strict dialogue structure with an explicit and inevitable initiative by the system (henceforth called *system–initiative* in contrast to *user–initiative*) results from the crucial fact that with any of these questions by the system a particular sub–grammar and sub–lexicon of the speech analysis system (e.g. a simple number or yes/no grammar and lexicon, respectively) can be associated to analyse the user's answer more reliably. Clarification dialogues caused by incorrectly analysed words can be circumvented by this method. Hence it is essential for a user–initiative or adaptive (or also called *mixed-initiative*) system to remedy the shortcomings resulting from the less reliable analysis of the user's spoken turn with a general grammar and lexicon, respectively. Furthermore, the *task parameters*, i.e. the information provided in the user's turn to perform the user–intended task by the system, have to be extracted without knowing exactly where in the user's turn or whether at all they have been uttered yet. In the case that not all task parameters are provided even a user–initiative system has to ask questions — similar to a system–initiative system.

KoHDaS–NN[1] is an *automatic help desk system in a call center* that basically leaves the dialogue initiative with the user as far as (s)he wants. The user's turn circumscribing the problem as a whole is handed to a hierarchy of recurrent plausibility networks which classify the according problem. In the next step the system extracts even only implicitly mentioned task parameters of this problem class from the turn by a graph-matching technique. Remaining or unidentified task parameters required to solve the problem are asked by the system in an ordinary question–answering manner. The results of KoHDaS–NN where the classification and the extraction of the task parameters is performed only on the basis of simple words are promising. However the number of wrong classifications and questions for yet uttered task parameters has to be further decreased in order to provide a user–friendly dialogue with the customers. Hence we investigate in the following whether deploying a shallow parser based on Supertagging increases the performance of the system — both with respect to the classification and the extraction of the task parameters.

The Supertagger in KoHDaS–ST relies on a *Hidden Markov model* and is trained with German input texts. The main section of this paper is devoted to the description of this method (cf. Section 3; it also comprises some

1. The acronym KoHDaS stands for **Ko**blenzer **H**elp **D**esk with automatic **S**peech recognition. In the following the basic version is called KoHDaS–NN (NN stands for **N**eural **N**etworks). Later on we investigate KoHDaS–ST where ST stands for **S**uper**T**agging.

implementation details to gain efficiency). With respect to accuracy our German Supertagger lags behind the English one. Some of the reasons will be addressed later on. Nevertheless shallow parsing with the Supertags increases the accuracy compared to a basic version of KoHDaS that only relies on recurrent plausibility networks.

The paper is organized as follows. In the next Section KoHDaS and its functionality is described. In Section 3 the Supertagger based on Hidden Markov models is outlined. Section 4 is devoted to the description of KoHDaS–ST, i.e. how Supertagging used in shallow parsing is integrated into the application of KoHDaS. Furthermore, it depicts the results of KoHDaS–ST compared to KoHDaS–NN. In Section 5 related work is portrayed. Here further methods that favorably compare to Supertagging are outlined on the one hand. On the other, different Supertagging methods and their application domains are delineated. The paper ends addressing future work and open problems.

2. KoHDaS — an adaptive dialogue system for First–Level Support via telephon

KoHDaS (see, e.g., (Harbusch, Knapp and Laumann, 2001)) explores methods which provide automatic user–initiative dialogues in call centers, i.e. the initiative should basically be left with the user as far as (s)he wants. Compared to system–initiative dialogue systems, user–initiative systems cannot rely on restricted dictionaries and grammars during the speech recognition process to gain more reliable results. Hence, methods to remedy the reduction of understanding in the first phase have to be impinged on the system.

KoHDaS–NN deploys the following two techniques towards a natural dialogue behaviour. First, a hierarchy of neural networks classifies the user's whole turn (on average 25 words) according to a list of problem classes. After a confirmation dialogue the *task parameters* mentioned in the user's turn are extracted to avoid redundant questions by the system for information to perform the task the user wants the system to perform — in our case a data base look–up for a solution of the user's hardware problem.

As for classification, a *hierarchy of recurrent plausibility networks* (cf. (Wermter, 1995)) accomplishes the consideration of arbitrary previous contexts in current decision making in KoHDaS. After a confirmation that the problem class is correctly recognized, the task parameters, i.e. the necessary information to enable the system to perform the task the user intended the system to do, are extracted from the user's initial turn by a graph matching technique imposed on the dialogue graphs, i.e. the specification of all possibly asked questions by the system and the according task parameters provided by the user (note that these graphs have to be specified anyhow to model a user who does not take the initiative at all).

KoHDaS is currently customized to the domain of first level support for computer hardware but it can easily be adapted to new domains.

A drawback of KoHDaS–NN is the absence of syntactic information in the processing of the user's turn as KoHDaS–NN works simply word–based. With respect to the classification task, a word such as *"monitor"* remains active by the context–layer independent whether the word in mentioned in a subordinate clause or as a key concept to characterize the problem (cf. *"My new monitor flickers since I have ..."* vs. *"My SCSI hard drive which I bought together with my new monitor two weeks ago from your customer's service cannot be formatted ..."*). With respect to the extraction of task parameters negations are completely ignored in KoHDaS–NN. For instance, *"My monitor flickers although the boxes are not activated"* takes for granted that the monitor only occasionally flickers when the boxes are activated. Hence the wrong conclusion is drawn from the customer's utterance. In order to remedy this shortcoming, KoHDaS–ST deploys a robust syntactic analysis based on a Supertagger (Joshi and Srinivas, 1994) and a Lightweight Dependency Analyzer (LDA) (Srinivas, 1997a) in the analysis of the user's turns. Particularly for the correct interpretation of the scope of negations we have decided to use Supertagging instead of chunking (cf. Section 5). The structural information — which possibly remains partial — provided by this analysis is used in the classification step as well as in the information extraction step of KoHDaS. The use of structural information in classification can prevent words occurring in deeply nested sub–sentences from having high impact on the determination of the problem class. In the information extraction step, the use of structural information allows to detect dependencies between structures in the turn, that can't be detected in the word–based version of KoHDaS–NN (e.g., negations; cf. Section 4).

3. Hidden Markov model–based Supertagging

Our German Supertagger uses a *trigram model* and is based on *Hidden Markov models (HMMs)* enabling the use of the well known algorithms on HMMs (see, e.g., (Rabiner, 1989)) to guess the most likely syntactic structure of a sentence. We use a trigram model as it has shown to achieve good results in Supertagging (cf. (Srinivas, 1997a)).

In this section the basic concepts of Hidden-Markov models are briefly introduced. Thereafter a Hidden Markov model–based Supertagger is portrayed. Finally, aspects of the implementation —particularly with respect to time and space efficiency — are highlighted.

3.1. Basic concepts of Hidden Markov models

Let us assume the following notational conventions (adapted from (Rabiner and Juang, 1986) and (Charniak, 1993) or see, e.g., (Rabiner, 1989) for a good introduction):

- T = length of the sequence of observations (training set),
- N = number of states (either known or guessed),
- M = number of possible observations (from the training set),
- $\Omega_X = \{q_1, ...q_N\}$ (finite set of possible states),
- $\Omega_O = \{v_1, ..., v_M\}$ (finite set of possible observations),
- X_t random variable denoting the state at time t (state variable),
- O_t random variable denoting the observation at time t (output variable),
- $\sigma = o_1, ..., o_T$ (sequence of actual observations)

and distributional parameters:

- $A = \{a_{ij}\}$ with $a_{ij} = Pr(X_{t+1} = q_j | X_t = q_i)$ (transition probabilities),
- $B = \{b_i\}$ with $b_i(k) = Pr(O_t = v_k | X_t = q_i t)$ (observation probabilities),
- $\pi = \{\pi_i\}$ with $\pi_i = Pr(X_0 = q_i)$ (initial state distribution).

A *Hidden Markov model (HMM)* is a five-tuple $(\Omega_X, \Omega_O, A, B, \pi)$. Let $\lambda = \{A, B, \pi\}$ denote the parameters for a given HMM with fixed Ω_X and Ω_O. This means, a discrete–time, discrete–space dynamical system governed by a Markov chain emits a sequence of observable outputs: one output (observation) for each state in a trajectory of such states. From the observable sequence of outputs, the most likely dynamical system can be inferred. The result is a model for the underlying process. Alternatively, given a sequence of outputs, the most likely sequence of states can be inferred. The model can also be used to predict the next observation or more generally a continuation of the sequence of observations. Three basic problems can be formulated for HMMs:

1. Find $Pr(\sigma|\lambda)$, i.e. the probability of the observations given the model.
2. Find the most likely state trajectory given the model and observations.
3. Adjust $\lambda = \{A, B, \pi\}$ to maximize $Pr(\sigma|\lambda)$.

For any of these questions efficient algorithms are known (see, e.g., (Rabiner, 1989)). The *Forward–Backward algorithm* (Baum and Eagon, 1967) solves the first problem, problem 2 is yielded by the *Viterbi algorithm* (Viterbi, 1967) und problem 3 can be dealt with by the *Baum–Welch algorithm* (Baum, 1972).

3.2. Hidden Markov models for Supertagging

Many variants of Supertagging use models similar to POS–Tagging (cf. Section 5 for a brief description of the variants Trigram Supertagging, Head Supertagging and Transformation–based Supertagging). Here, we underlay the Supertagger with *Hidden Markov models (HMMs)*.

In this framework, the Supertags are encoded as states and the words as symbols of the output alphabet of the HMM. Assuming a bigram model (i.e. *n–Gram* with $n = 2$), the realization is easy and straightforward. Any Supertag becomes an individual state and any terminal an individual output symbol. The according Tagger can be trained with the Baum–Welch algorithm. The observation sequence is provided by the sentences of the training set (unsupervised learning). However, this method lacks behind supervised learning methods (see, e.g., (Merialdo, 1994)). Such a corpus can be gained with the *Viterbi algorithm* (cf. problem 2 mentioned above), as this algorithm denotes the optimal sequence of states for a given observation sequence — in our case the optimal sequence of Supertags.

The Supertagger we report on in this paper uses a trigram model (cf. (Bäcker, 2001) for an evaluation of the HMM–based Supertagger using bigrams). According to the trigram model, two previous states (i.e. Supertags) are encoded in the HMM in a well–known manner (see, e.g., (El-Beze and Merialdo, 1999)):

- The states of the HMM correspond to pairs of Supertags (t_{i-1}, t_i).
- The transition probability $Pr[(t_{i-1}, t_i)|(t_{i-2}, t_{i-1})]$ is denoted by the trigram probability $Pr(t_i|t_{i-2}t_{i-1})$.
- The output symbols are provided by the words which are tagged with t_i and which are emitted in states $(\_, t_i)$.

At the beginning of a sentence pseudo states $(\emptyset, t_j)$ with $\emptyset$ a pseudo category are assumed.

In general, the Baum–Welch algorithm (cf. problem 3) can be applied to optimize the model parameters in order to maximize $Pr(\text{training set}|\lambda)$. Our results are gained on the basis of a labeled corpus. Hence we don't impose the Baum–Welch algorithm on our Supertagger. On the basis of the labeled corpus we directly estimate the model parameters according to the *Maximum Likelihood Estimation (MLE)* method. For trigrams this means:

$$Pr_{MLE}(t_i|t_k, t_j) = \frac{c(t_k, t_j, t_i)}{c(t_k, t_j)}, \quad 1 \leq i \leq N, \quad 1 \leq j \leq N, \tag{1}$$

$$Pr_{MLE}(w_k|t_j) = \frac{c(w_k, t_j)}{c(t_j)}, \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \tag{2}$$

with:

$c(t_j)$	$\hat{=}$	number of occurrences of t_j in the training set,
$c(t_j, t_k)$	$\hat{=}$	number of occurrences of t_j followed by t_k,
$c(t_k, t_j, t_i)$	$\hat{=}$	number of occurrences of t_k followed by t_j, which itself is followed by t_i, and
$c(w_k, t_j)$	$\hat{=}$	number of occurences of w_k labelled as t_j.

In order to overcome problems with *sparse data*, i.e. not all trigrams occur in the training set, *smoothing techniques* (for a good introduction see, e.g., (Jurafsky and Martin, 2000) or (Manning and Schütze, 2000); in (Chen and Goodman, 1996) the performance of various smoothing techniques are evaluated) are applied in our system. Furthermore the treatment of unknown words is described in the following.

In our system we employ *Good–Turing Discounting* (Good, 1953). This means, the equation (1) of the MLE estimation which relies on the absolute frequency $c(t_k, t_j, t_i)$ of a trigram is replaced by the following equation:

$$Pr_{GT}(t_i|t_k, t_j) = \frac{c^*(t_k, t_j, t_i)}{c(t_k, t_j)}, \quad 1 \leq i \leq N, \quad 1 \leq j \leq N, \tag{3}$$

with c^* the modified number of trigrams. The Good–Turing Discounting computes c^* according to:

$$c^* = c \quad \text{für}\, c > k \tag{4}$$

and according to (Katz, 1987):

$$c^* = \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}} \quad \text{for}\, 1 \leq c \leq k. \tag{5}$$

Here the constant k denotes a threshold to prevent the system from re–estimating rather accurate results (according to high frequency).

The discounting method can be further improved by a method that differentiates between unseen trigrams. The *Backoff method* (Katz, 1987) uses the frequencies of $(n-1)$–grams in the following manner:

$$Pr_{BO}(t_i|t_{i-2}t_{i-1}) = \begin{cases} \tilde{P}(t_i|t_{i-2}t_{i-1}), & \text{if}\, c(t_{i-2}t_{i-1}t_i) > 0, \\ \alpha_1 \tilde{P}(t_i|t_{i-1}), & \text{if}\, c(t_{i-2}t_{i-1}t_i) = 0 \text{ and } c(t_{i-1}t_i) > 0, \\ \alpha_2 \tilde{P}(t_i), & \text{otherwise.} \end{cases} \tag{6}$$

If the frequency of a trigram (bigram, resp.) is zero, the frequency of the bigram (unigram, resp.) is considered. However, a factor α_1 (α_2, resp.) is supposed to normalize the resulting probability. This means, for any given t_n the following holds:

$$\sum_{i,j} Pr(t_n|t_i t_j) = 1. \tag{7}$$

In formula (6) $\tilde{P}$ denotes the probability which results form a discounting method (otherwise the values don't fulfil equation (7)). We use again the Good–Turing discounting method here. The according formula looks as follows:

$$Pr_{BO}(t_i|t_{i-2}t_{i-1}) = \begin{cases} Pr_{GT}(t_i|t_{i-2}t_{i-1}), & \text{if } c(t_{i-2}t_{i-1}t_i) > 0, \\ \alpha(t_{n-2}^{n-1})Pr_{GT}(t_i|t_{i-1}), & \text{if } c(t_{i-2}t_{i-1}t_i) = 0 \text{ and } c(t_{i-1}t_i) > 0, \\ \alpha(t_{n-1})Pr_{GT}(t_i), & \text{otherwise.} \end{cases} \tag{8}$$

Unknown words are treated in our system in the following manner. The probability $Pr(w_k|t_j)$ is computed by the Backoff method. In case w_k is an unknown word we adapt the method by (Weischedel *et al.*, 1993) which deals with *features* of words. The prefixes and suffixes of words are considered to estimate the probabilities according to the following formula:

$$Pr(w_k|t_j) = \begin{cases} Pr_{MLE}(w_k|t_j) & \text{if } c(w_k, t_j) > 0, \\ Pr(unknown|t_j) * Pr(features|t_j) & \text{otherwise.} \end{cases} \tag{9}$$

The probability of the occurrence of an unknown word $Pr(unknown|t_j)$ for the currently considered Supertag is estimated according to:

$$Pr(unknown|t_j) = \frac{N_1(t_j)}{c(t_j)}, \tag{10}$$

where $N_1(t_j)$ is the number of words which occur in the training set exactly once with the Supertag t_j; $Pr(features|t_j)$ denotes the probability whether a word with the same prefix or suffix as w_k, respectively, occurs together with the Supertag t_j.

Now we face the task of tagging itself. The tagging is performed by the Viterbi algorithm. For a given observation sequence $O = \{O_1, O_2, \ldots, O_T\}$ the most likely sequence of states $Q = \{q_1, q_2, \ldots, q_T\}$ is computed in four steps (Initialization, Recursion, Termination and Reconstruction (Path Backtracking); the time complexity of the Viterbi algorithm ist $O(N^2 n)$ where n is the length of the input). For a German test set of 30 sentences, 78.3% of the words were assigned the correct Supertag. In the conclusions we compare this result with English Supertagging.

In general the above described HMM–bases Supertagger was tested with a German corpus of 250 tagged sentences (cf. the evaluation in Section 4). The German training and test corpus has been constructed in the following manner. Basically we looked at written German dialogues in news groups in the area of first level support for computer hardware. We developed an LTAG with 127 elementary trees covering the domain of the KoHDaS system (cf. (Bäcker, 2002)) and automatically parsed these dialogues. The reviewed results of all parses constitute the tagged corpus. We trained our Supertagger using 250 tagged sentences. For the estimation of the HMM's model parameters we used Good–Turing discounting combined with Katz's Backoff model to smooth the parameters resulting in better estimations of unseen trigrams. We use word features similar to (Weischedel *et al.*, 1993) (i.e. prefixes and suffixes of words) to estimate the observation probability of unknown words.

3.3. Implementation of the HMM–based Supertagger

The overall system is implemented in Java. In this paragraph we highlight some implementational details which reduce space and time complexity of our system (cf. (Cutting *et al.*, 1992) for a discussion of efficiency matters for POS-Taggers).

Let us first beer in mind which complexity a HMM comes along with. The model parameters of a HMM consist of N states and M output symbols from a $N \times N$ matrix A of transition probabilities, a $N \times M$ matrix B

of observation probabilities and a N–dimensional vector π (initial state distribution). All these parameters have to be yielded, i.e. the space complexity is $O(N^2 + MN)$.

The states of our HMM comprise pairs of Supertags. Hence the number of states equals the square of the number of Supertags T. Consequently the space complexity is $O(T^4 + MT^2)$, und the run time of the Viterbi algorithm is $O(T^4 n)$. From this fact directly follows that the model parameters cannot be represented by a two–dimensional array (for the 127 Supertags in our system, the two–dimensional array of 64–bit digits for the transition probabilities requires 2 GB space). As a consequence, all model parameters are stored in an *associative* manner in our system [2].

A reasonable space reduction results from only storing probabilities greater than zero[3]. With respect to the transition probabilities the following holds. These probabilities are computed during the training phase where they don't become smoothed. Smoothing is performed during tagging. During that process the trigram, bigram and unigram models are determined. Furthermore, the factors of the Backoff method are computed. A smoothed probability is only computed on demand (`getA(`(t_{i-2}, t_{i-1})`)`). Consequently the overall space complexity depends on the actually deployed training set (unknown trigrams are not stored).

With respect to the run time the following improvements can be performed to gain more efficiency in Supertagging. The Viterbi algorithm iterates over all words and all states in the following manner:

```
for each word $w_i$ in sentence {
    for each state $m$ {
        for each state $n$ {
            ⋮
        }
        ⋮
    }
}
```

Shortcuts for states with an observation probability zero and unique POS can reduce the run time reasonably. The associative hash tables allow to access all states of the currently considered word occurring in the training phase. These sets computed for the current word and its predecessor build the basis to collect the set of *relevant states* of the current word (Backoff method for the observation probabilities). Only for these states the iteration needs to be performed instead of the nested iteration over all states. More formally speaking:

$$\texttt{relevantStates}(i,j) = \{(t_k, t_l) \mid Pr(w_i|t_k) > 0 \wedge Pr(w_j|t_l) > 0\}$$

is supposed to be regarded in the two nested loops mentioned above. For $i < 0$ and $j < 0$, respectively, w_i and w_j, respectively, denote the pseudo words at the beginning of a sentence. Assuming only relevant states decreases the average run time reasonably. Our Supertagger requires approximately 28 ms for the tagging of a sentence only conducting relevant states whereas it runs at least a second if all states are considered.

4. Application of Supertagging in the user–initiative Help Desk system KoHDaS

The results of the Supertagger described in the previous section allow a LDA (Srinivas, 1997a) to discover dependencies between the Supertags in the user's turn. The dependency structure accomplished by this method is used in classification and in information extraction in the following manner.

In KoHDaS–NN, the user's turn is classified according to *significance vectors* of the form:

$$v(w, c_i) = \frac{\text{occurrences of a word from } w \text{ within class } c_i}{\sum_{j=1}^{n} \text{occurrence of words from } w \text{ within class } c_j}$$

2. The associative storing in Java is realized by the class HashMap, which provides a Hash table (see, e.g., (Flanagan, 2002)).

3. It is important to notice here that due to the fact that the real–digit arithmetics cannot differentiate between zero an very small values (as holds for the products of probabilities computed in the Viterbi algorithm) we deal with the logarithms of the probabilities in the hash table of the probabilities of Supertags. This states a suitable method here because not probabilities themselves but the arguments of such probabilities are maximized, i.e. the Viterbi algorithm computes the maximum sum of the logarithms of the probabilities instead of the maximum product of the probabilities.

where only 616 words are actually regarded and matched with a reduced vocabulary with 131 word groups c_i, i.e. general concepts in our domain (such as *'hard disk'*, *'monitor'* and *'capacity'*) containing words, which can be considered to be synonymous (e.g., words in class *'hard disk'* are *'disk'* or *'harddrive'*). Generally, significance vectors account for the importance of the word in a specific problem class.

In KoHDaS–ST, these significance vectors are adjusted using the results of the structural information collected by the Supertagger and the LDA. The adjusted significance vectors v^* are computed by:

$$v^*(w, c_i) = \begin{cases} \alpha^d\, v(w, c_i) & \text{if } v(w, c_i) > \frac{1}{n}\sum_{j=1}^{n} c_j, \\ (1 + \beta d) * 0.1 & \text{if } v(w, c_i) = 0 \text{ and } d > 0, \\ (1 + \beta d)\, v(w, c_i) & \text{otherwise.} \end{cases}$$

where d represents the syntactic depth of the sentence in that the current word occurs and α and β are constant values. Tests have shown that suitable values for α and β are $\alpha = 0.8$ and $\beta = 0.6$.

The results of this approach compared to the pure neural net–based version of KoHDaS are outlined in Table 1. The table shows that the Mean Squared Error (MSE) of three sub–networks of KoHDaS–NN is decreased in the top level net as well as in the local net for disk problems but increased in the local net for monitor problems. The reasons why the monitor problems behave in this unexpected manner are topic of future investigations (cf. Section 6).

Table 1: Mean Squared Error (MSE).

Net	MSE in Test	
	KoHDaS–NN	KoHDaS–ST
NN differentiating monitor and disk probs.	3.85	3.45
Local NN - disk probs.	10.33	9.65
Local NN - monitor probs.	4.72	4.91

In the graph–based information extraction step, each node of the graph corresponds to the information already extracted from the turn. Nodes can be associated with questions to be asked by the system. Edges are labeled with sets of word groups enabling a transition if an appropriate word occurs in the user's input. See Figure 1 for a partial dialogue graph of KoHDaS–NN.

In KoHDaS–ST the results of the dependency analysis together with features in the lexicon are used to create a kind of *semantic representation* of the user's turn (cf. (Bäcker, 2002)). Edges in the new dialogue graphs are labeled with this representation (see Figure 2) resulting in improved processing of the turn. For example the turn *"Sometimes my computer drives me mad. My monitor started glimmering 3 days ago."* would enable the transition from node 5 to 51, as *'sometimes'* is found in the input. In KoHDaS–ST this will not happen, because *'sometimes'* is not related to *'glimmering'*.

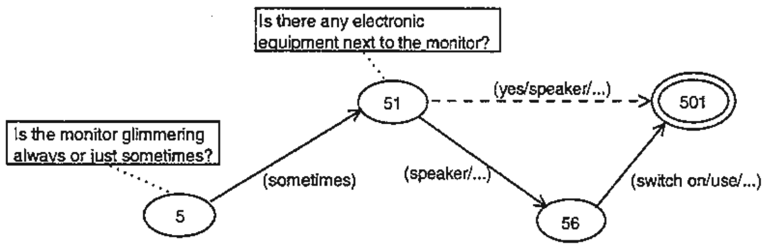
Is there any electronic equipment next to the monitor?
Is the monitor glimmering always or just sometimes?
51
(yes/speaker/...)
501
(sometimes)
(speaker/...)
(switch on/use/...)
5
56

Figure 1: Part of a dialogue graph in the problem class of glimmering monitors in KoHDaS.

5. Related work

A well studied method to extract relevant information from potentially ill–formed (as is the case for spoken utterances) or not completely mastered (as is the case for automatically analysed spoken utterances) input is *chunk*

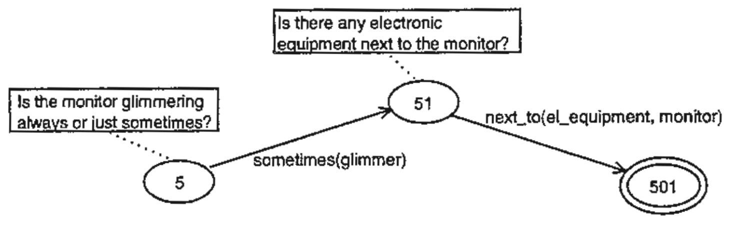
Is there any electronic equipment next to the monitor?
Is the monitor glimmering always or just sometimes?
51
5
501
sometimes(glimmer)
next_to(el_equipment, monitor)

Figure 2: Part of a dialogue graph in the problem class of glimmering monitors in KoHDaS–ST.

parsing (also called *chunking*; see, e.g., (Abney, 1991), (Appelt *et al.*, 1993), (Grishman, 1995) or (Hobbs *et al.*, 1997))). Cascaded finite state automata analyse various phrase types expressed in terms of regular expressions. The main advantage of this approach is its robustness and the fast run time. The main obstacle of chunking issues from the restricted formal power of finite state automata. Cascades of several levels (cf. the system FASTUS (Appelt *et al.*, 1993), (Hobbs *et al.*, 1997) with five levels) allow for the analysis of recursive structures to some extend. The basic level accepts smaller linguistic constructions, whereas in the next levels these elements become grouped into larger linguistic units. Accordingly, FASTUS can basically recognize 'complex words' such as proper nouns consisting of several individual words. However the coverage remains restricted to the static number of cascaded phases.

Another robust and fast parsing method offers *statistical parsing*. According to a labeled corpus generalization rules can be extracted (cf. *Treebank*, (Marcus, Santorini and Marcinkiewicz, 1993)). These rules can be *grammar rules* (see, e.g., (Charniak, 1997), (Collins, 1996)) or *decision trees* (see, e.g., (Magerman, 1995)). Each rule becomes associated with a probability, which is determined in the training phase with respect to the corpus. Parsing means to find the most likely derivation according to these rules. The term statistical parsing subsumes a further variant where the rule set is also determined beforehand (see, e.g., (Black *et al.*, 1993)). In the training phase probabilities for these rules are computed according to the corpus (cf. *Probabilistic Context–Free Grammars* (*PCFG*, (Booth, 1969)). The goal of this method is primarily disambiguation whereas robustness is a not so relevant here.

Let us finally mention recent work in the area of Supertagging. Supertagging (Joshi and Srinivas, 1994) is a disambiguation method which extends *Part–Of–Speech Tagging* (*POS–Tagging*). A *Lexicalized Tree Adjoining Grammar*, i.e. any rule has at least one lexical anchor (cf. (Schabes, 1990)) underpins the system. As is the case for Part–Of–Speech Tagging, the model is trained with a labeled corpus. In the training phase, each word of each input sentence becomes associated with a lexicalized rule according to the model (*Supertag*). On the basis of this relation, a Lightweight Dependency Analyzer (LDA, (Srinivas, 1997a)) identifies relations between the Supertags of the individual lexical entries of a input sentence, i.e. the grammar rules the Supertagger gives the highest probabilities. As far as possible a complete parse is computed. Particularly the ability to produce complete parses (if possible) compared to individual phrases in chunking led to the choice of the latter method for our application domain. Here, the user's utterance should be best analysed particularly in the step of extracting task parameters. Our conjecture is that Supertagging allows for a more elaborate identification of complex constructions such as negations and their scope in the user's utterance.

In the area of Supertagging various approaches for the model have been proposed in the literature (e.g., *Trigram Supertagging* by (Srinivas, 1997a) with Good–Turing Discounting (Good, 1953) and the Backoff method by Katz (Katz, 1987)). On the basis of a training corpus of 1 000 000 English words the Supertagger provides an accuracy of 92,2%. *Head Trigram Supertagging* (Srinivas, 1997a) is a similar method based on trigrams. However not the two previous Supertags are used to compute the current Supertag but the two previous *Head Supertags*. A Head Supertag is a previously computed Supertag which influences the choice of the current Supertag. The method works in two phases. In the first one all Head Supertags are determined. In the second phase, the Head Supertags are used to compute the probabilities of all Supertags. This method assigns in 91,2% of the cases the correct Head Supertags for a training corpus of 1 000 000 words. Its accuracy is 87%.

Transformation–based Supertagging (Srinivas, 1997a), (Srinivas and Joshi, 1999) adapts the central idea of transformation–based POS–Tagging (Brill, 1993), (Brill, 1995). For this method any word in the corpus is labeled with its most frequent tag. During Tagging these tags can be changed by a list of transformations. Such a transformation consists of a pattern, which activates the rule and a rewriting rule. In order to train this Supertagger a set of transformation patterns and a labeled reference corpus has to be provided. The training algorithm determines the best order of rule applications by minimizing the error rate of the Tagger compared to the reference corpus. The Supertagger based on this model has been trained with 200 000 words and reaches an accuracy of 90%.

A Supertagger for German (Bäcker, 2001) based on Hidden Markov models and a bigram model was trained and tested with word classes instead of individual words. Notice that from this fact a loss of accuracy results. Furthermore only basic smoothing techniques were imposed. Accordingly, first results lack far behind the previously mentioned ones. On the basis of 5 460 sentences of the NEGRA corpus (Brants *et al.*, 1997) the Supertagger has an accuracy of 65,5%. This was basically the reason to impose trigrams to the Supertagger we describe here.

Supertagging and Lightweight Dependency Analyzers exhibit high robustness and efficiency[4] and are deployed for shallow parsing in various contexts (e.g., *text chunking* with Supertags (Srinivas, 1997b)). Text chunking (Abney, 1991) means that a sentence is divided into several non-overlapping segments. By this method individual types of phrases (e.g., identification of noun phrases *Noun Chunking*) can be identified in a text. A respective LDA reaches high precision (91,8%) and recall (93%) (cf. (Srinivas, 1997b)).

6. Conclusions

In order to conclude, the application of Hidden Markov model–based Supertagging in the user–initiative dialogue system KoHDaS–ST helps to remedy the lack of accuracy resulting from speaker–independent speech recognition on the basis of general dictionaries and grammars as required in a user–initiative dialogue system.

Comparing the results of German Supertagging (78.3%) to English (92.2%), two different reasons lead to less good results. First, our training set (1 973 words) is small compared to the English one (1 000 000 words). Accordingly, many unseen trigrams are imposed on the system. Second, German is a language with free word order. This fact amplifies the effects of sparse data (cf. Spanish Supertagging has an accuracy of about 80% (Srinivas, 1998)). In the future the training set of KoHDaS–ST will be extended. Furthermore, unsupervised learning methods integrated with supervised methods (cf. (Montoya, Suárez and Palomar, 2002)) will be deployed in our system. How far we can get with a free word order language like German is currently an open problem.

A further open problem is why the adjustment of significance vectors according to the identified sentence structure decreases the number of correctly classified problems with respect to the class of monitor problems. Perhaps our conjecture that concepts mentioned in subordinate clauses have a lower impact to the decision making than those in the main clause is too strict.

References

Abney, Steven. 1991. Parsing by chunks. In Robert Berwick, Steven Abney and Carol Tenny, editors, *Principle-based parsing: Computation and Psycholinguistics*. Kluwer Academic Publishers, Dortrecht, The Netherlands, pages 257–278.

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David Israel and Mabry Tyson. 1993. FASTUS: A Finite-state Processor for Information Extraction from Real-world Text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI–93)*, pages 1172–1178, Chambéry, France.

Bäcker, Jens. 2001. Entwicklung eines Supertaggers für das Deutsche. Studienarbeit, Universität Koblenz-Landau, Institut für Computerlinguistik, Koblenz, Germany.

Bäcker, Jens. 2002. KoHDaS–ST — Supertagging in dem automatischen Dialogsystem KoHDaS. Diplomarbeit, Universität Koblenz-Landau, Institut für Computerlinguistik, Koblenz, Germany.

Baum, Leonard E. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8.

Baum, Leonard E. and J. A. Eagon. 1967. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of American Mathematical Society*, 73:360–363.

Black, Ezra, Frederick Jelinek, John Lafferty, David M. Magerman, Robert Mercer and Salim Roukos. 1993. Towards History–based Frammars: Using Richer Models for Probabilistic Parsing. In *Proceedings of the 31st Conference of the Association of Computational Linguistics (ACL–93)*, pages 31–37, Columbus, Ohio, USA.

Boje, Johan, Mats Wiren, Manny Rayner, Ian Lewin, David Carter and Ralph Becker. 1999. Language–Processing Strategies and Mixed–Initiative Dialogues. In J. Alexanderson, L. Ahrenberg, K. Jokinen and Jönsson, editors, *Special Issue on Intelligent Dialogue Systems. ETAI (Electronic Transactions on Artificial Intelligence).*

Booth, Taylor L. 1969. Probabilistic representation of formal languages. In *IEEE Conference Record of the 10th Annual Symposium on Switching and Automata Theory*, pages 74–81.

Brants, Thorsten, Roland Hendriks, Sabine Kramp, Brigitte Krenn, Cordula Preis, Wojciech Skut and Hans Uszkoreit. 1997. Das NEGRA-Annotationsschema. Negra project report, Universität des Saarlandes, Computerlinguistik, Saarbrücken, Germany.

Brill, Eric. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL–93)*, pages 259–265, Columbus, Ohio, USA.

4. The run time of a Supertagger–based LDA is $O(n)$ where n denotes the length of the input.

Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–566.

Charniak, Eugene. 1993. *Statistical Language Learning*. Cambridge, Massachusetts: MIT Press.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI–97)*, pages 47–66, Menlo Park, CA, USA.

Chen, Stanley F. and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL–96)*, pages 310–318.

Collins, Michael John. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL–96)*, pages 184–191, San Francisco, CA, USA.

Cutting, Douglas, Julian Kupiec, Jan O. Pedersen and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP)*, pages 133–140, Trento, Italy.

El-Beze, Marc and Bernard Merialdo, 1999. "Hidden Markov Models". In Hans van Halteren, editor, *Syntactic wordclass tagging*, chapter 16, pages 263–284. Dordrecht, the Netherlands: Kluwer Academic Publishers.

Flanagan, David. 2002. *Java in a Nutshell*. 4th edition. Cambridge, MA, USA: O'Reilly.

Good, Irving J. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264.

Grishman, Ralph. 1995. The NYU System for MUC-6 or Where's the Syntax? In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 167–175, San Francisco, CA, USA.

Harbusch, Karin, Melanie Knapp and Christoph Laumann. 2001. Modelling user-initiative in an automatic help desk system. In Hitoshi Isahara and Qing Ma, editors, *Proceedings of the Second Workshop on Natural Language Processing and Neural Networks (NLPNN2001)*, pages 69–76, Tokyo, Japan.

Hobbs, Jerry R., Douglas E. Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel and Mabry Tyson. 1997. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In Emmanuel Roche and Yves Schabes, editors, *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, MA, USA, pages 383–406.

Joshi, Aravind K. and Bangalore Srinivas. 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING–94)*, pages 154–160, Kyoto, Japan.

Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing*. Upper Saddle River, NJ, USA: Prentice Hall.

Katz, Slava M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401.

Litman, Diane, Shimei Pan and Marilyn Walker. 1998. Evaluating response strategies in a web–based soken dialogue agent. In *Proceedings of the 36th Annual Meeting of the ACL and the 17th International Conference on Computational Linguistics (COLING–ACL–98)*, pages 780–786, Montreal, Canada.

Magerman, David M. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL–95)*, pages 276–283, Cambridge, MA, USA.

Manning, Christopher D. and Hinrich Schütze. 2000. *Foundations of statistical language processing*. Cambridge, MA, USA: MIT Press.

Marcus, Mitchell P., Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Merialdo, Bernard. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

Montoya, Andrés, Armando Suárez and Manuel Palomar. 2002. Combining Supervised–Unsupervised Methods for Word Sense Disambiguation. In Alexander Gelbukh, editor, *Proceedings of the 3rd International Conferences on Intelligent Text Processing and Computational Linguistics (CICLING)*, pages 156–164, Mexico City, Mexico. Springer.

Rabiner, Lawrence R. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, Lawrence R. and Biing-Hwang Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January.

Schabes, Yves. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Srinivas, Bangalore. 1997a. *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Srinivas, Bangalore. 1997b. Performance evaluation of Supertagging for partial parsing. In *Proceedings of Fifth International Workshop on Parsing Technology (IWPT–97)*, Boston, USA.

Srinivas, Bangalore. 1998. Transplanting Supertags from English to Spanish. In *Proceedings of Fourth International Workshop on Tree-Adjoining Grammars (TAG+4)*, pages 5–8, Philadelphia, PA, USA.

Srinivas, Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Strachan, Linda, John Anderson, Murrey Sneesby and Mark Evans. 1997. Pragmatic user modelling in a commercial software system. In *Proceedings of the 6th International Conference on User Modeling*, pages 189–200, Chia Laguna, Italy.

Viterbi, Andrew J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269.

Weischedel, Ralph, Marie Meteer, Richard Schwartz, Lance Ramshaw and Jeff Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382.

Wermter, Stefan. 1995. *Hybrid connectionist natural language processing*. London, Great Britain: Chapman and Hall, International Thomson Computer Press.