Applications of Statistical Machine Translation Approaches to Spoken Language Understanding

Klaus Macherey, Oliver Bender, and Hermann Ney, *Senior Member, IEEE*

***Abstract*—In this paper, we investigate two statistical methods for spoken language understanding based on statistical machine translation. The first approach employs the source-channel paradigm, whereas the other uses the maximum entropy framework. Starting with an annotated corpus, we describe the problem of natural language understanding as a translation from a source sentence to a formal language target sentence. We analyze the quality of different alignment models and feature functions and show that the direct maximum entropy approach outperforms the source channel-based method. Furthermore, we investigate how both methods perform if the input sentences contain speech recognition errors. Finally, we investigate a new approach to combine speech recognition and spoken language understanding. For this purpose, we employ minimum error rate training which directly optimizes the final evaluation criterion. By combining all knowledge sources in a log-linear way, we show that we can decrease both the word error rate and the slot error rate. Experiments were carried out on two German inhouse corpora for spoken dialogue systems.**

***Index Terms*—Combined approach, machine translation, maximum entropy, minimum error rate training, speech recognition, spoken language understanding.**

I. Introduction

In spoken language understanding (SLU), the objective is to extract all the information from a speech-based input that is relevant to a specific task. Formally, the objective can be described as follows: given an input utterance represented as a sequence of acoustic vectors $x_1^T = x_1, \ldots, x_T$, we search for the sequence of concepts $\hat{c}_1^M = \hat{c}_1, \ldots, \hat{c}_M$ having the highest probability among all possible concept sequences

$$\hat{c}_1^M = \underset{c_1^M, M}{\operatorname{argmax}} \left\{ pr\left(c_1^M | x_1^T\right) \right\} \tag{1}$$

$$= \underset{c_1^M, M}{\operatorname{argmax}} \left\{ \sum_{w_1^N, N} \underbrace{pr\left(c_1^M | w_1^N\right)}_{\text{NLU}} \cdot \underbrace{pr\left(w_1^N | x_1^T\right)}_{\text{ASR}} \right\} \tag{2}$$

where the word sequence $w_1^N = w_1, \ldots, w_N$ is a hidden variable and $pr(c_1^M | w_1^N)$ is assumed to be independent of the sequence of acoustic vectors. $pr(w_1^N | x_1^T)$ denotes the statistical approach to automatic speech recognition (ASR) and $pr(c_1^M | w_1^N)$ is the statistical natural language understanding (NLU) component. Often, (2) is approximated by a sequential decoding

$$\hat{c}_1^M = \underset{c_1^M, M}{\operatorname{argmax}} \left\{ pr\left(c_1^M | \hat{w}_1^N\right) : \hat{w}_1^N = \underset{w_1^N, N}{\operatorname{argmax}} \left\{ pr\left(w_1^N | x_1^T\right) \right\} \right\} \tag{3}$$

that is, in the first step, the optimal word sequence $\hat{w}_1^N$ maximizing $pr(w_1^N | x_1^T)$ is determined which is then passed to the NLU component in the second step. This approximation can be suboptimal because errors made in the ASR component often remain undetected and are propagated to the subsequent NLU component. To alleviate this problem, the optimal word sequence $\hat{w}_1^N$ can be replaced by word graphs or N-best lists on which then the search for the optimal concept sequence $\hat{c}_1^M$ is carried out [1].

Modeling the conditional probability $pr(c_1^M | w_1^N)$ is a complex task. Many NLU systems perform a rule-based semantic analysis using stochastic context free grammars in order to determine the optimal sequence of concepts [2]. Although rule-based approaches are useful if only little or no data is available, providing handcrafted, consistent rules is often very costly. Therefore, alternative approaches are useful that reduce the manual effort of writing rules explicitly. In this paper, our main focus is on modeling the NLU probability and on combining ASR with NLU such that the overall error criterion is minimized. The NLU models we propose are based on statistical machine translation (SMT). Throughout the paper, we will make use of ideas and concepts used in SMT and apply them to the problem of NLU. The remainder of this paper is organized as follows. In Section II, we give a general view of NLU based on SMT from which we derive two approaches, a source channel-based approach and a maximum entropy (ME)-based approach. We describe the concept language used and introduce alignment models employed for both approaches. In Section III, we describe the source-channel approach using phrase-based machine translation. In Section IV, we define several feature functions that are suitable for NLU and are used in the ME model. The framework for combining ASR with NLU is outlined in Section V. Related work is discussed in Section VI and results are presented in Section VII.

II. General View of Natural Language Understanding

The objective of NLU is to extract all the information from a natural language-based input that is relevant to a specific task.

Manuscript received May 21, 2008; revised November 23, 2008. Current version published April 01, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ruhi Sarikaya.

K. Macherey was with the Computer Science Department 6, RWTH Aachen University, D-52056 Aachen Germany. He is now with Google, Inc., Mountain View, CA 94043 USA (e-mail: kmach@google.com).

O. Bender and H. Ney are with the Human Language Technology and Pattern Recognition, Computer Science Department 6, RWTH Aachen University, D-52056 Aachen, Germany (e-mail: bender@cs.rwth-aachen.de; ney@cs.rwth-aachen.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2009.2014262

Often, stochastic grammars are used for this task requiring handcrafted rules, that is, dependencies between words and concepts must be modeled explicitly. Although partial parsing techniques are well suited to parse ungrammatical sentences, the sole usage of rule-based methods can turn out to be inflexible. Especially, when extending the application scenario or the application domain itself, many rules must be rewritten by hand to adjust them to new phrases and keywords. Therefore, the question is how this process can be simplified and whether the complex dependencies between words and concepts can be learned automatically by just providing pairs of sentences and concept strings.

To achieve this goal, we take a closer look at the NLU problem. Basically, NLU can be viewed as a translation from a source language to a formal target language, that is, given a natural language source sentence $w_1^N = w_1, \ldots, w_N$ we choose the formal language target sentence $\hat{c}_1^M = \hat{c}_1, \ldots, \hat{c}_M$ among the space of all possible translations that has the highest probability, leading to the following decision rule:

$$\hat{c}_1^M = \underset{c_1^M, M}{\operatorname{argmax}} \left\{ pr\left(c_1^M | w_1^N\right) \right\} \tag{4}$$

$$= \underset{c_1^M, M}{\operatorname{argmax}} \left\{ pr\left(c_1^M\right) \cdot pr\left(w_1^N | c_1^M\right) \right\}. \tag{5}$$

Equations (4) and (5) induce two different approaches: either we can directly optimize the posterior probability $pr(c_1^M|w_1^N)$ using, for example, the maximum entropy framework or we can employ the classical source-channel approach using Bayes' theorem and decompose the posterior probability into two probability distributions: the translation probability $pr(w_1^N|c_1^M)$ and the concept language probability $pr(c_1^M)$. Once we have substituted these probability distributions with suitable model distributions, we can learn the model parameters with supervised learning techniques. Before we describe the methods in more detail we will first discuss the concept language used and review some basic ideas from machine translation like alignment models that we employ for both approaches in order to automatically align words to concepts.

A. Concept Language

A concept c is defined as the smallest unit of meaning that is relevant to a specific task [3]. Concept representations are sometimes used in a hierarchical manner, that is, concepts can be nested and subsume other concepts [4]. Here, we use nonhierarchical concepts which we call *flat* concepts. By using flat concepts as formal target language, the annotation of the input sentences can easily be provided because for a human annotator no structural knowledge about concepts and their interplay is necessary. Although not a strict requirement, we assume that the concepts occur in the same order as given by the source text. An example is provided in Fig. 1. The links between the sentences indicate to which concepts the words are aligned. Note that these alignments between words and concepts are not given explicitly but must be determined automatically from the training data which is provided as pairs of source sentences and concept strings. Although not depicted in the figure, each concept is associated with a (possibly empty) set of attributes and each attribute is associated with a more or less complex rule describing how to extract the value from the words aligned to the concept.

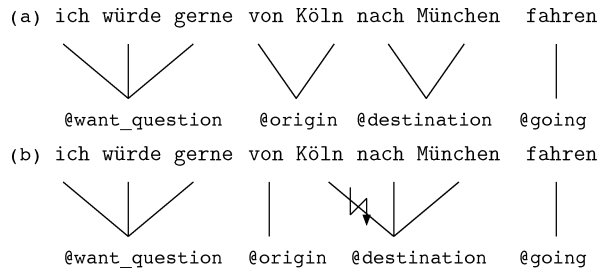

Fig. 1. Example of a correctly and wrongly aligned word sequence for the utterance "I would like to go from Cologne to Munich." In (a), the phrase "from Cologne" is correctly aligned to the concept *@origin*. In (b), the city name "Cologne" is wrongly aligned to the concept *@destination*. If each concept together with its associated words is considered separately, it will be impossible to extract the attribute value for the concept *@origin* in case (b).

B. Alignment Models and Local Context Dependencies

As depicted in Fig. 1, the meaning of a word and thus its alignment to the correct concept depends on its context. For example, the allocation of the city name *Köln (Cologne)* to the concept *@origin* is only apparent when looking at the preceding preposition *von (from)* in the source sentence. Although the sequence of concepts is identical in both cases, only case (a) allows for a correct extraction of the attribute values. If each concept together with its aligned words is considered independently from all the other concepts, it will be impossible to extract the correct attribute value from the concept *@origin* in case (b). Therefore, finding correct alignments automatically is essential for both approaches. In the following, we will briefly review different alignment models that are used in machine translation and which we will use for either approach. A consequence of words and concepts having the same order is that the generated alignments will be monotonic. Also, we enforce that every word is aligned to exactly one concept. This might be problematic if the source language contains compounds that could be aligned to more than one concept. For example, the German compound "Privatadresse (home address)" as opposed to employer's address could be aligned to two target concepts *@private* and *@address* to distinguish it from "Büroadresse (office address)" which could be mapped to *@office* and *@address*. To enforce the one-word-to-one-concept constraint, we can design the target language appropriately and shift the distinction between home and office address into the attributes of the *@address* concept. Words not contributing to the meaning of a sentence are aligned to filler concepts.

For the introduction of the alignment models, we switch our notation and identify a word sequence w_1^N with f_1^J and a sequence of concepts c_1^M with e_1^I in order to use the same notation that is common in machine translation. If we rewrite the translation probability $pr(f_1^J|e_1^I)$ by introducing a hidden alignment

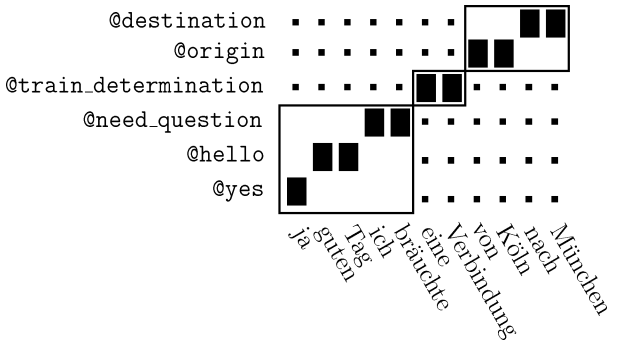

Fig. 2. Phrase-based translation with alignment templates for the utterance "yes, hello, I need a connection from Cologne to Munich." The words on the x-axis denote the source words, the words on the y-axis are the concepts. The rectangles mark phrase boundaries; the solid boxes within the rectangles describe word-to-concept alignments.

$a_1^J = a_1, \ldots, a_j, \ldots, a_J$, with $a_j \in \{1, \ldots, I\}$, we obtain the following identity:

$$\begin{aligned} & pr\left(f_1^J|e_1^I\right) \\ &= \sum_{a_1^J} pr\left(f_1^J, a_1^J|e_1^I\right) \qquad (6) \\ &= pr\left(J|e_1^I\right) \cdot \sum_{a_1^J} \prod_{j=1}^{J} pr\left(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I\right) \qquad (7) \\ &= \underbrace{pr\left(J|e_1^I\right)}_{\text{length model}} \cdot \sum_{a_1^J} \prod_{j=1}^{J} \underbrace{pr\left(a_j|a_1^{j-1}, f_1^{j-1}, e_1^I\right)}_{\text{alignment/distortion model}} \\ &\quad \cdot \underbrace{pr\left(f_j|f_1^{j-1}, a_1^j, e_1^I\right)}_{\text{lexicon/translation model}} . \qquad (8) \end{aligned}$$

The different alignment models we consider result from different decompositions of $pr(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I)$. In the following, we will briefly introduce these alignment models which were proposed in [5] and [6], and are the basis for extracting word-concept phrase pairs.

1) HMM Alignment: If we assume a first-order dependence for the alignment a_j in (8), restrict the dependent quantities of the translation probability only to a_j, and assume a simple length model $pr(J|e_1^I) = p(J|I)$, we arrive at the HMM alignment [6]

$$pr\left(f_1^J|e_1^I\right) = p(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} \left[p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j})\right] . \qquad (9)$$

2) Model 1: By replacing the dependency in (9) from a_{j-1} to j, we obtain a zero-order hidden Markov model. Assuming a uniform alignment probability $p(i|j, I) = \frac{1}{I}$, we obtain the much simpler model 1 which we use as starting point

$$pr\left(f_1^J|e_1^I\right) = \frac{p(J|I)}{I^J} \cdot \sum_{a_1^J} \prod_{j=1}^{J} \left[p\left(f_j|e_{a_j}\right)\right] . \qquad (10)$$

3) Model 3, Model 4, and Model 5: In Models 3 and 4, fertilities of target words are introduced. The fertility of a target word is the number of source words it can generate. Models 3-5 are alignment models with increasing complexity. Their exact definitions can be found in [5] but are not important for the basic ideas presented in this paper. Starting from model 1, the sequential application of the more complex alignment models together with a search process finally yields the Viterbi alignment a_1^J between words and concepts.

III. Natural Language Understanding Using the Source-Channel Paradigm

For the source channel-based approach, we use a phrase-based translation system based on so-called alignment templates [7]. Due to $a_j \in \{1, \ldots, I\}$, the basic alignment models introduced in the previous section can capture only 1-to-many alignments. A phrase-based translation system goes one step further and allows many-to-many alignments by providing a two-level alignment: a phrase level alignment and a within-phrase many-to-many word-level alignment. The key idea for obtaining many-to-many word alignments is to symmetrize the training directions. By performing a training in both translation directions (from source to target and from target to source) we obtain two Viterbi alignments a_1^J and b_1^I for each sentence pair which can be merged. Although 1-to-many alignments are sufficient because of the restriction that each word must align to exactly one concept, alignments obtained from symmetrization and merging have usually a higher precision [8]. Let the source and target sentence be segmented into K word-groups describing the phrases

$$\begin{aligned} e_1^I &= \widetilde{e}_1^K, \widetilde{e}_k = e_{i_{k-1}+1}, \ldots, e_{i_k}, \quad k = 1, \ldots, K \\ f_1^J &= \widetilde{f}_1^K, \widetilde{f}_k = f_{j_{k-1}+1}, \ldots, f_{j_k}, \quad k = 1, \ldots, K. \end{aligned}$$

By decomposing the translation probability with the above-mentioned definitions, we arrive at the following first-order decomposition:

$$pr\left(f_1^J|e_1^I\right) = \sum_{\widetilde{a}_1^K, K} \prod_{k=1}^{K} p(\widetilde{a}_k|\widetilde{a}_{k-1}) \cdot p\left(\widetilde{f}_k|\widetilde{e}_{\widetilde{a}_k}\right). \qquad (11)$$

With $z = (\widetilde{e}', \widetilde{f}', \widetilde{a}')$ denoting an alignment template, we obtain $p(\widetilde{f}_k|\widetilde{e}_{\widetilde{a}_k}) = \sum_z p(z|\widetilde{e}_{\widetilde{a}_k}) \cdot p(\widetilde{f}_k|z, \widetilde{e}_{\widetilde{a}_k})$. The phrase translation probability $p(\widetilde{f}_k|z, \widetilde{e}_{\widetilde{a}_k})$ is then decomposed according to the following equation:

$$\begin{aligned} & p\left(\widetilde{f}_k|(\widetilde{e}', \widetilde{f}', \widetilde{a}'), \widetilde{e}_{\widetilde{a}_k}\right) = \\ & \qquad \delta\left(\widetilde{e}_{\widetilde{a}_k}, \widetilde{e}'\right) \cdot \delta(\widetilde{f}_k, \widetilde{f}') \cdot \prod_{j=j_{k-1}+1}^{j_k} p\left(f_j|\widetilde{a}', \widetilde{e}_{\widetilde{a}_k}\right). \qquad (12) \end{aligned}$$

The Kronecker functions $\delta(\cdot, \cdot)$ ensure that only those alignment templates z are chosen which are consistent with $\widetilde{f}_k$ and $\widetilde{e}_{\widetilde{a}_k}$. The probability $p(f_j|\widetilde{a}', \widetilde{e})$ can be decomposed in the following

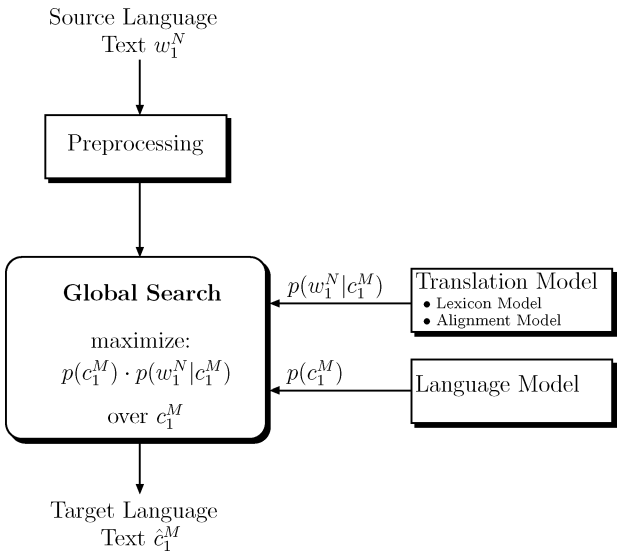

Fig. 3. Architecture of the natural language understanding component using a statistical machine translation approach based on the source-channel paradigm.

way:

$$p(f_j|\widetilde{a}',\widetilde{e}) = \sum_{i=1}^{I} p(i|j;\widetilde{a}') \cdot p(f_j|e_i) \tag{13}$$

$$p(i|j;\widetilde{a}') = \frac{\widetilde{a}'(i,j)}{\sum_{i'} \widetilde{a}'(i',j)}, \tag{14}$$

$$\widetilde{a}'(i,j) := \begin{cases} 1, & \text{if } (i,j) \text{ are linked in the merged alignment} \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Fig. 2 depicts an example of a phrase-based translation using alignment templates.

A. Training

In machine translation, source words that have no correspondence on the target side are mapped onto the empty word. In contrast to translations between natural languages, we do not allow empty words, that is, each word of the source sentence must be aligned to a concept of the target sentence since words that do not contribute to the meaning of a sentence are explicitly modeled by a filler concept. The same holds if we exchange source and target language. This results in the following training procedure.

1) The fertility for the empty word is explicitly set to 0.
2) We compute the Viterbi alignment for each sentence of the training corpus and symmetrize the training directions [8].
3) We compute the alignment matrices using intersections with refinements as combination method [8].

B. Search

According to (5), both probability distributions are combined during a search process which leads to the architecture depicted in Fig. 3. If we plug in the phrase-based translation model together with a standard left-to-right trigram language model into the source-channel approach, we obtain the following search criterion in maximum approximation which is used in combination with beam search:

$$\hat{e}_1^I = \underset{e_1^I, I}{\operatorname{argmax}} \left\{ pr\left(e_1^I\right) \cdot pr\left(f_1^J|e_1^I\right) \right\} \tag{16}$$

$$= \underset{e_1^I, I}{\operatorname{argmax}} \left\{ \prod_{i=1}^{I} p\left(e_i|e_{i-2}^{i-1}\right) \cdot \max_{\widetilde{a}_1^K, \widetilde{z}_1^K, K} \left\{ \prod_{k=1}^{K} \cdot p(\widetilde{a}_k|\widetilde{a}_{k-1}) \cdot p(z_k|\widetilde{e}_{\widetilde{a}_k}) \cdot p\left(\widetilde{f}_k|z_k, \widetilde{e}_{\widetilde{a}_k}\right) \right\} \right\}. \tag{17}$$

IV. Natural Language Understanding Using Maximum Entropy

Alternatively to the source-channel approach, we can directly model the posterior probability $pr(c_1^M|w_1^N)$ in (4). A well-founded framework for doing this is maximum entropy (ME). This framework allows for integrating information from many heterogeneous information sources for classification. The data for a classification problem is described as a number of problem-specific features $h_l, l = 1, \ldots, L$. Each feature h_l corresponds to a constraint of the model and is associated with a model parameter λ_l. Among all possible models that satisfy these constraints, the model with maximum entropy is computed during the training phase [9]. Within this framework, the posterior probability can be modeled as follows:

$$pr\left(c_1^M|w_1^N\right) = p_{\lambda_1^L}\left(c_1^M|w_1^N\right) = \frac{\exp\left[\sum_{l=1}^{L} \lambda_l h_l\left(c_1^M, w_1^N\right)\right]}{\sum_{c'^M_1} \exp\left[\sum_{l=1}^{L} \lambda_l h_l\left(c'^M_1, w_1^N\right)\right]}. \tag{18}$$

The architecture of the ME approach is depicted in Fig. 4. The feature functions in (18) depend on complete sentences. Similar to [10], we could generate a set of candidate translations and apply the ME framework on top of this candidate set, so that the λ_l parameters can be estimated such that we obtain better translations. This step would require a separate decoder to generate the candidate translations which we want to avoid. If we can formulate the NLU task directly within the ME framework we will not depend on an additional classifier generating lists of candidate translations, but would define feature functions on the basis of words and phrases and apply them directly on the source sentences. Furthermore, we do not just want to combine the knowledge sources from the previous section in a log linear way, but also want to overcome some shortcomings of the alignment templates approach. A shortcoming of the alignment templates approach is that each word can be used in at most one template. Overlapping templates are not allowed due to the segmentation into K word groups. For the ME approach, we still want to assign each word to one concept, but we also want to use the context of each word in a more flexible way such that each word can be used in more than one feature function. To solve this problem in the ME approach, we proceed as follows. We slightly modify the NLU task such that source and target sentence are of the same length ($N = M$) yielding a one-to-one

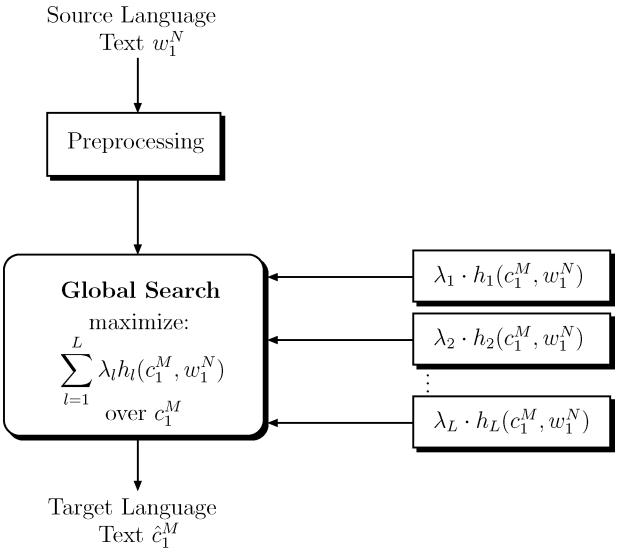

Fig. 4. Architecture of the natural language understanding component using a statistical machine translation approach based on the maximum entropy framework.

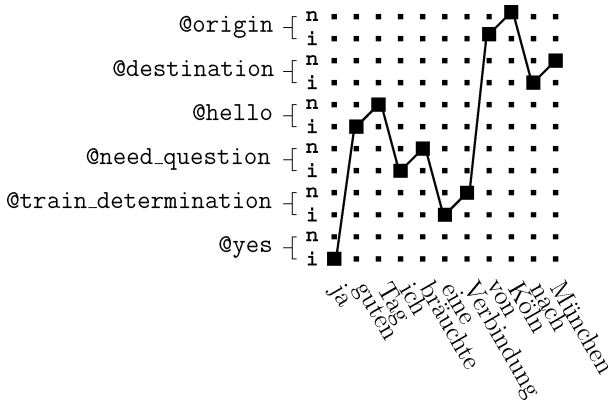

Fig. 5. Example of a sentence-to-concept mapping for the utterance "yes, hello, I need a connection from Cologne to Munich" using maximum entropy. Each concept starts with an initial concept marked by "i" and optionally continues with a noninitial concept denoted by "n."

correspondence between words and concepts.[1] As discussed in Section II-B, mapping compounds to concepts is not a problem. A case that occurs more frequently is that several contiguous words are mapped onto a single concept. Therefore, we distinguish whether a word belongs to an initial or a noninitial concept. This modeling was explicitly required in shallow parsing tasks [11]. In the context of NLU, it was introduced in [12]. With these changes, our classification model becomes very similar to maximum entropy Markov models (MEMMs) which are often used in information extraction tasks [13]. The difference to a standard information extraction problem is that for the NLU task described in this paper, we do not know the correct alignment between words and concepts during training, that is, word and concept sequences have different lengths. Thus, MEMMs are not directly applicable here. We solve this problem by first computing the Viterbi alignment between words and concepts on our training data and then by keeping the alignment fixed for all subsequent training steps. Fig. 5 depicts a one-to-one mapping applied to a sentence/concept pair from the German TABA corpus. We assume that the decisions only depend on a window $w_{n-2}^{n+2} = w_{n-2} \dots w_{n+2}$ around word w_n and on the predecessor concept. Thus, we can factorize the posterior probability of (4) and obtain the following first-order model:

$$pr\left(c_1^N|w_1^N\right) = \prod_{n=1}^{N} pr\left(c_n|c_1^{n-1}, w_1^N\right) \tag{19}$$

$$= \prod_{n=1}^{N} p_{\lambda_1^L}\left(c_n|c_{n-1}, w_{n-2}^{n+2}\right). \tag{20}$$

Due to the distinction between initial and noninitial concepts, we must ensure that a noninitial concept only follows its corresponding initial one which can be guaranteed by appropriate transition features.

[1]Because a concept can cover multiple words, this will turn the notion of a concept into a fractional concept which we call a *concept tag*. Thus, a consecutive sequence of similar concept tags then forms a concept. For ease of terminology we will continue to speak of concepts rather than of tags.

A. Feature Functions

In the following, we will describe a set of binary valued feature functions that are used for text-based NLU. We will introduce additional feature functions when we apply the ME framework on speech-based inputs.

1) Lexical Features: The lexical feature $h_{w,d,c}$ employs lexical information. The parameter d allows for taking context words into account. Formally, the feature is defined as follows:

$$h_{w,d,c}\left(c_{n-1}^n, w_{n-2}^{n+2}\right) = \delta(w_{n+d}, w) \cdot \delta(c_n, c) \quad d \in \{-2, \dots, 2\}. \tag{21}$$

It will only fire if the word w_{n+d} matches w and if the prediction for the current concept c_n is equal to c. Here, $\delta(\cdot,\cdot)$ denotes the Kronecker function.

2) Prefix and Suffix Features: In case that a test corpus contains words that were not observed in the training corpus (unknown words), the necessity for vocabulary independent features arises. To achieve this, we generalize the words by looking at their prefixes or suffixes. Let $w_n = \alpha\beta$ be a decomposition of word w_n such that α is a prefix of w_n and β is its suffix. If the prefix (suffix) of w_n is equal to a given prefix (suffix), these features will fire:

$$h_{\overline{\alpha},c}\left(c_{n-1}^n, w_{n-2}^{n+2}\right) = \begin{cases} 1, & \text{if } \exists \alpha, \beta : w_n = \alpha\beta \wedge \alpha = \overline{\alpha} \wedge c_n = c \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

$$h_{\overline{\beta},c}\left(c_{n-1}^n, w_{n-2}^{n+2}\right) = \begin{cases} 1, & \text{if } \exists \alpha, \beta : w_n = \alpha\beta \wedge \beta = \overline{\beta} \wedge c_n = c \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

3) Capitalization Features: A capitalization feature will fire if w_n starts with a capitalized letter, has an internal capital letter,

or is all capitalized

$$h_{\mathrm{cap},c}\left(c_{n-1}^{n}, w_{n-2}^{n+2}\right) = \begin{cases} 1, & \text{if } w_n \text{ starts with a capital letter } \wedge\, c_n = c \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

$$h_{\mathrm{int},c}\left(c_{n-1}^{n}, w_{n-2}^{n+2}\right) = \begin{cases} 1, & \text{if } w_n \text{ has an internal capital letter } \wedge c_n = c \\ 0, & \text{otherwise.} \end{cases} \quad (25)$$

$$h_{\mathrm{all},c}\left(c_{n-1}^{n}, w_{n-2}^{n+2}\right) = \begin{cases} 1, & \text{if } w_n \text{ is all capitalized } \wedge\, c_n = c \\ 0, & \text{otherwise.} \end{cases} \quad (26)$$

Capitalization features can be used to detect new proper names that were not observed during training.

4) Transition Features: Transition features model the dependence on the predecessor concept

$$h_{c',c}\left(c_{n-1}^{n}, w_{n-2}^{n+2}\right) = \delta(c_{n-1}, c') \cdot \delta(c_n, c). \quad (27)$$

This feature will fire if the prediction c_n of the current class is equal to c and the predecessor concept c_{n-1} is equal to c'.

5) Prior Features: The single concept priors are incorporated by prior features. The prior feature h_c describes the prior knowledge of the concept c on the training corpus and is the concept's unigram count

$$h_c\left(c_{n-1}^{n}, w_{n-2}^{n+2}\right) = \delta(c_n, c). \quad (28)$$

6) Compound Features: The feature functions defined so far produce only features that refer to single words or concepts. To enable word phrases and word-concept combinations, we introduce the following compound features

$$h_{\{z_1,d_1\},\ldots,\{z_K,d_K\},c}\left(c_{n-1}^{n}, w_{n-2}^{n+2}\right) = \prod_{k=1}^{K} h_{z_k,d_k,c}\left(c_{n-1}^{n}, w_{n-2}^{n+2}\right), \quad z_k \in \{w, c'\}, d_k \in \{-2,\ldots,2\} \quad (29)$$

where K defines the length of a compound feature. For $z_k = w$ the compound feature is constructed of lexical features. For $z_k = c'$ it corresponds to multiple transition features with the additional constraint that $d \in \{-1, 0\}$.

B. Training

The convexity of the ME objective function prohibits the incorporation of hidden variables. Thus, it is not possible to incorporate a hidden alignment as we did in Section III. Therefore, we will determine the alignment beforehand and keep it fixed during training. For training, we consider the set R of aligned training sentences to form a single long sentence. Since models trained within the ME framework tend to overfit on training data, we follow [14] and apply a Gaussian prior $p(\lambda_1^L)$ on the parameter set. This yields the following training criterion:

$$\hat{\lambda}_1^L = \underset{\lambda_1^L}{\operatorname{argmax}} \left\{ p\left(\lambda_1^L\right) \sum_{r=1}^{R} \sum_{n=1}^{N_r} p_{\lambda_1^L}\left(c_n | c_{n-1}, w_{n-2}^{n+2}\right) \right\} \quad (30)$$

where

$$p\left(\lambda_1^L\right) = \prod_{l=1}^{L} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{\lambda_l^2}{2\sigma^2}\right] \quad (31)$$

with the standard deviation σ as the smoothing parameter.

C. Search

For decoding a sequence of words w_1^N, we perform a breadth-first search where all hypotheses are expanded position-synchronously, that is, we start with expanding hypotheses for word w_1 at position 1, then continue with hypotheses for word w_2 at position 2, etc. To solve the search problem in maximum approximation (*Viterbi search*) we define the auxiliary quantity $Q_{\max}(n, c)$ as the maximal probability of the best partial path covering positions $1, \ldots, n$ and ending in concept c. This leads to the following recursive equations which are solved using dynamic programming

$$Q_{\max}(0, \$) = 1 \quad (32)$$

$$Q_{\max}(n, c) = \max_{c'} \left\{ Q_{\max}(n-1, c') \cdot p_{\lambda_1^M}\left(c | c', w_{n-2}^{n+2}\right) \right\}. \quad (33)$$

Here, \$ denotes the sentence start symbol. If no pruning is applied, the Viterbi search is guaranteed to find the most likely sequence of concepts w.r.t. the underlying models.

V. Combining Speech Recognition and Natural Language Understanding

As discussed in Section I, the sequential application of ASR and NLU can turn out to be suboptimal. Therefore, we need an approach that allows for a tighter coupling between both components. Furthermore, the approach used should allow us to easily integrate additional knowledge sources. The ASR module is based on the source-channel paradigm. Since it is difficult to integrate additional knowledge sources into this framework, we will use N-best lists for representing the most likely word hypotheses that are promising candidates for the NLU task, and perform all following investigations on this structure. Using N-best lists rather than word graphs has the advantage that also sentence-based features can be employed. The features will be combined in a log-linear model and the corresponding feature weights are trained in a discriminative manner using the minimum error rate criterion. We will proceed as follows: we briefly review the minimum error rate criterion and describe the optimization of the feature weights λ_1^L using the Line Sweep algorithm. The optimization is an iterative procedure that is applied on an N-best repository. As error criterion to be minimized we will use both the slot error rate and the word error rate. Thus, we can investigate whether for a given set of feature functions the NLU component can benefit from ASR features and, vice versa, whether the ASR component can benefit from features derived from the NLU component.

A. Minimum Error Criterion

Minimum error criteria have a wide range of applications in pattern recognition tasks. In the field of ASR, a minimum error criterion was for example employed in [15]. In the context of

machine translation, it was first used in [16]. To the best of our knowledge minimum error criteria have not been applied to SLU tasks. Since we regard NLU as a translation task, we can define the minimum error criterion similar to the approach proposed in [16] with the extension that we aim at minimizing both the word error rate and the slot error rate. The challenging task is then to find appropriate feature functions that result in a reduction of both error rates.

Denote $\mathbf{x} = x_1^T$ the sequence of acoustic observation vectors, $\mathbf{w} = w_1^N$ the sequence of words, and $\mathbf{c} = c_1^M$ the sequence of concepts, then we want to find that sequence of parameters $\hat{\lambda}_1^L$ which minimizes the number of errors denoted by an error function $E(\cdot,\cdot)$

$$\hat{\lambda}_1^L = \underset{\lambda_1^L}{\operatorname{argmin}} \left\{ \sum_{s=1}^{S} E\left((\mathbf{c}_s, \mathbf{w}_s), (\hat{\mathbf{c}}, \hat{\mathbf{w}})\left(\mathbf{x}_s; \lambda_1^L\right)\right) \right\} \tag{34}$$

where

$$(\hat{\mathbf{c}}, \hat{\mathbf{w}})\left(\mathbf{x}_s; \lambda_1^L\right) = \underset{(\mathbf{c},\mathbf{w})\in\mathcal{R}_s}{\operatorname{argmax}} \left\{ \sum_{l=1}^{L} \lambda_l h_l(\mathbf{x}_s, \mathbf{w}, \mathbf{c}) \right\}. \tag{35}$$

Here, s denotes the sentence index and $(\hat{\mathbf{c}}, \hat{\mathbf{w}})$ denotes the maximizing concept and word sequence, respectively. $\mathcal{R}_s$ defines the repository of candidates which for each utterance s is the product of the N-best candidates from the ASR module times the $\mathcal{N}$-best list of concept strings for each recognized sentence candidate. For ease of terminology, we will refer to pairs $(\mathbf{c}, \mathbf{w})$ of translation and sentence hypotheses as candidate translations. Once produced by the two decoding steps they are kept fixed throughout the algorithm.

B. Line Sweep Algorithm

Algorithm 1: Minimum Error Rate Training

Input: initial weights λ_1^L

Output: optimized weights $\hat{\lambda}_1^L$

repeat

Generate N-best repositories with current λ_1^L

for all dimensions l **do**

for all sentences s **do**

Compute upper envelope and error statistics

end for

Merge error statistics

Search for optimal γ and determine error reduction Δe_l

$\lambda_l' \leftarrow \lambda_l + \gamma$

end for

$\lambda_l \leftarrow \lambda_l'$ with $\hat{l} = \underset{l}{\operatorname{argmin}}\{\Delta e_l\}$

until convergence.

The sum of weighted feature functions can be expressed as a scalar product

$$\sum_{l=1}^{L} \lambda_l h_l(\cdot,\cdot,\cdot) = \left(\lambda_1^L\right)^\top \cdot \left(h_1^L\right). \tag{36}$$

In each iteration we try to change the weight vector λ_1^L such that the total number of errors is minimized. This is an L-dimensional optimization problem. For simplicity, we loop over all dimensions and optimize each dimension separately. Let γ denote the change of weight λ_l for a given dimension l. Then this change can be expressed as

$$g_l(\gamma) : \left(\lambda_1^L + \gamma \cdot d_1^L\right)^\top \cdot \left(h_1^L\right) = \underbrace{\left(d_1^L\right)^\top \left(h_1^L\right)}_{\text{slope } a} \cdot \gamma + \underbrace{\left(\lambda_1^L\right)^\top \left(h_1^L\right)}_{\text{offset } b} \tag{37}$$

where (d_1^L) is the L-dimensional unit vector with component $l = 1$. For each utterance $\mathbf{x}_s$ there is a repository $\mathcal{R}_s$ of candidate translations. Each candidate translation defines a line g_l with slope a and offset b, and is associated with its error w.r.t. the reference translation. For the one-dimensional optimization problem, the goal is now to adjust parameter λ_l such that the error is minimal. From a geometric point of view the intersection points of the lines divide each line into at most $|\mathcal{R}_s|$ line segments. The line segments form polyhedrons partitioning the plane into regions of equal numbers of errors. However, due to the arg max decision rule we only need to consider those line segments with maximal scores, that is, which belong to the *upper envelope*. A line segment belongs to the upper envelope if a vertical upward ray starting at this line segment does not hit any other line. Since each line is associated with its error rate we can search for that segment along the upper envelope which minimizes the error. By projecting the points of intersections of the corresponding line segments together with their error counts onto the x-axis, we obtain a histogram of the total error counts. The phase transitions mark the positions where the total error may change. The basic principle is depicted in Fig. 6.

The upper envelope can be efficiently computed using the *line sweep* algorithm which was introduced in [17]. A vertical line is moved from the far left to the far right of the plane, and the exact order of the intersections with the segments is maintained as the line sweeps along. If we compute the error counts for all sentences and merge the projected points of intersections together with their error counts, we obtain a finite sequence of λ_l values together with their associated error change. A linear search in this sequence yields the optimal parameter λ_l. The pseudo code for this procedure is given in Algorithm 1 which is guaranteed to converge to a local optimum.

C. Feature Functions

We introduce a new set of sentence-based feature functions in order to take ASR and NLU features into account. The ASR N-best lists are annotated with concepts using the ME approach. We omit the one-to-one correspondence between words and initial-continue concepts and use the (c_1^M, w_1^N) notation, where w_1^N denotes an ASR sentence hypothesis stored in an N-best list, and c_1^M denotes a candidate translation from the

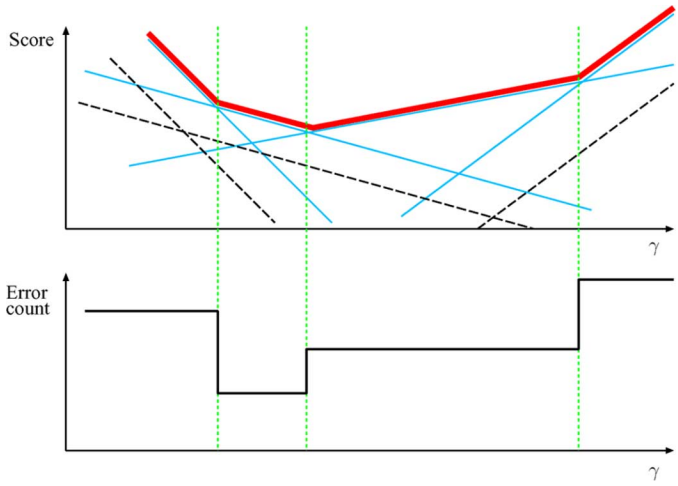

Fig. 6. Upper envelope and error counting. The bold red line segments define the upper envelope of the set of solid blue lines. The dashed black colored lines are parallel lines that have a lower score than the blue lines and can be discarded for the computation of the upper envelope. Each segment of the upper envelope defines an area of constant errors. By projecting the upper envelope to the x-axis, we obtain a histogram of the total number of errors. The total number of errors can only change at phase transitions, that is, at the points of intersections of the line segments.

NLU module stored in a corresponding $\mathcal{N}$-best list, that is, each recognition result stored in an N-best list produces a set of $\mathcal{N}$-best translations. Thus, a user utterance is represented by a total of $N \times \mathcal{N}$ sentence pair hypotheses. For ease of notation we will not distinguish between N and $\mathcal{N}$, but rather use the term N-best list. Using these conventions, we can define the following sentence-based feature functions.

1) Language Model Features: The language model feature assigns each candidate sentence derived from an N-best list its language model probability. Here, we use trigram language models and define feature functions for both class-based and non class-based language models. Thus, we obtain the following two feature functions:

$$h_{\mathrm{LM}}\left(c_1^M, w_1^N, x_1^T\right) = \prod_{n=1}^{N} p\left(w_n | w_{n-2}^{n-1}\right) \tag{38}$$

and

$$h_{\mathrm{classLM}}\left(c_1^M, w_1^N, x_1^T; \mathcal{K}\right) = \prod_{n=1}^{N} p(w_n|k_n) \cdot p\left(k_n | k_{n-2}^{n-1}\right) \tag{39}$$

where $\mathcal{K}$ denotes the set of word classes k for the class-based language model.

2) Acoustic Model Features: Similar to the language model features we use the acoustic score of a sentence hypothesis as additional feature. This yields the following feature function:

$$h_{\mathrm{AM}}\left(c_1^M, w_1^N, x_1^T\right) = \prod_{n=1}^{N} p\left(x_\tau^t | w_n\right) \tag{40}$$

where x_τ^t denotes the sequence of acoustic vectors assigned to word hypothesis w_n.

3) ASR Sentence Length: The length of an ASR sentence hypothesis is defined as the number of evaluation words contained in this sentence, disregarding silence and noise words. The weight for the sentence length is usually referred to as word penalty. Formally, the feature function is defined as follows:

$$h_{\mathrm{Length_{ASR}}}\left(c_1^M, w_1^N, x_1^T\right) = N. \tag{41}$$

4) ASR Posterior Probabilities: Word posterior probabilities were introduced in [18]. By defining the posterior probability of a sentence hypothesis as the product of its word posterior probabilities we arrive at

$$h_{\mathrm{Post_{ASR}}}\left(c_1^M, w_1^N, x_1^T\right) = \prod_{n=1}^{N} p\left([w_n; \tau, t] | x_1^T\right). \tag{42}$$

5) ASR Confidence Measures: Word posterior probabilities are also the basis for word-based confidence measures. For a sentence-based feature function, we define the geometric mean of the word-based confidence values $\tilde{p}$ contained in a sentence hypothesis as the sentence confidence value. This yields the following feature function:

$$h_{\mathcal{C}_{\mathrm{ASR}}}\left(c_1^M, w_1^N, x_1^T\right) = \left(\prod_{n=1}^{N} \tilde{p}\left([w_n; \tau, t] | x_1^T\right)\right)^{1/N}. \tag{43}$$

6) NLU Model Features: The NLU model feature is the factorized posterior probability obtained from the ME model defined in (20) for a given input sentence w_1^N. This is the first feature function that takes NLU-related knowledge sources into account. It is defined as follows:

$$h_{\mathrm{NLU}}\left(c_1^M, w_1^N, x_1^T\right) = \prod_{m=1}^{M} p_{\lambda_1^L}\left(c_n | c_{n-1}, w_{n-2}^{n+2}\right). \tag{44}$$

7) NLU Sentence Length: Similar to the ASR sentence length we use the NLU sentence length as an additional feature in order to penalize sentence hypotheses that are too short or too long

$$h_{\mathrm{Length_{NLU}}}\left(c_1^M, w_1^N, x_1^T\right) = M. \tag{45}$$

8) Concept Language Model: A trigram language model trained on concepts is used as additional feature function. The concept language model uses absolute discounting with backing off as smoothing variant. The feature function is defined as follows:

$$h_{\mathrm{LM_{NLU}}}\left(c_1^M, w_1^N, x_1^T\right) = \prod_{m=1}^{M} p\left(c_m | c_{m-2}^{m-1}\right). \tag{46}$$

VI. Related Work

The first approach to NLU using methods derived from machine translation is described in [19]. The approach uses a variant of model 1 and lacks local context dependencies. In

[10], this approach is extended by employing the ME framework for reranking the hypotheses of the system output.

Using generative models for NLU has been widely investigated. In [3], the NLU component is based on an HMM-like process where the hidden states correspond to flat concepts. In [20], tree-structured concepts are employed. The hidden vector state model introduced in [21] associates each state of a push-down automaton with the state of an HMM and is complex enough to capture hierarchical structures. More recently, discriminative methods have been applied. In [22], support vector machines are used for a call classification task which is extended in [23] using Adaboost.

Another active research field is how to obtain a tighter coupling between ASR and NLU. In [24], finite-state transducers are employed for combining ASR with translation models. Because NLU can be viewed as a translation task, this approach is directly applicable to SLU. In [25], the language model is replaced by a word-to-concept translation model which is directly integrated into an ASR framework. Other researchers have enriched the NLU component with ASR knowledge sources. In [26], ASR-based confidence measures are incorporated into the understanding process. In [27], word-level confusion networks obtained from ASR word graphs are exploited and combined with confidence measures in order to reduce the classification error rate of a dialog system. The contributions of this paper are made clear after the results are presented.

VII. Results

All experiments were conducted on two corpora, the TELDIR and the TABA corpus. The TELDIR corpus is a German in-house telephone directory assistance corpus. The objective is to answer naturally spoken requests for telephone numbers, fax numbers, and email addresses of persons, companies, and organizations. The corpus also covers utterances for spelling units. The data was recorded over several months from fixed-line phones as well as wireless and mobile phones under changing conditions. The conditions cover clean speech, office noise, and traffic noise. The corpus allocation is summarized in Table I. In contrast to the ASR vocabulary, the NLU source vocabulary is open.

The TABA corpus is a German in-house corpus covering the domain of a train timetable scheduling task. The speech data was collected from real users interacting with a fully functional automatic spoken dialogue system developed by Philips Speech Processing [28]. The TABA corpus was kindly provided by Philips Speech Processing and Scansoft. The corpus allocation is summarized in Table II. Similar to the TELDIR corpus, each utterance of the TABA corpus is annotated with a flat conceptual meaning representation. Again, the ASR vocabulary is closed whereas the NLU vocabulary is open.

The ASR component for both tasks is based on maximum likelihood-trained Gaussian mixture models with a pooled variance. We use 6-state HMMs in Bakis topology for modeling triphones where two contiguous states are identical. An acoustic feature vector consists of 12 MFCCs with first and second derivatives, resulting in a 25–dimensional feature vector. An LDA is applied which maps three consecutive acoustic feature vectors onto a final 33-dimensional feature vector. The ASR decoder employs an integrated trigram Viterbi beam search. The system is optimized for real-time recognitions in a spoken dialogue system. For acoustic adaptation, we use an incremental F-MLLR which estimates the transformation matrix on previous utterances from a user for a single dialogue transaction. The system generates word graphs and N-best lists under real-time constraints. Table III lists recognition and confidence measure results for both tasks. The ASR confidence measures are computed according to [18]. The *confidence error rate* (CER) is the ratio of the sum of word insertion and

TABLE I
CORPUS ALLOCATION FOR THE TELDIR CORPUS. TELDIR IS A GERMAN IN-HOUSE CORPUS COVERING THE DOMAIN OF A TELEPHONE DIRECTORY ASSISTANCE TASK. WITHIN THIS TASK, USERS CAN ASK FOR TELEPHONE NUMBERS, FAX NUMBERS, AND EMAIL ADDRESSES OF PERSONS, COMPANIES, AND ORGANIZATIONS

corpus TELDIR-Speech	training	development	evaluation
amount of acoustic data [h]	2:00	0:36	0:28
silence portion [%]	58.4	60.7	62.7
# speakers	161	49	41
# sentences	1 399	405	308
# running words	8 840	2 575	1 950
# running phonemes	36 390	10 572	8 040
covered vocabulary	883	443	369
# lexicon entries	1 348		
# pronunciations	1 403		
# unknown words	–	0	0
perplexity (trigram LM)	5.2	22.8	18.7
perplexity (class-trigram LM)	16.4	23.1	22.6

corpus TELDIR-NLU	training		development		evaluation	
	src	trg	src	trg	src	trg
# sentences	1 399		405		308	
# running words	8 840	3 564	2 575	1 054	1 950	801
covered vocabulary	883	22	443	21	369	19
# singletons	0	1	–	–	–	–
# unknown words	–	–	257	0	168	0
perplexity (trigram LM)	5.2	4.1	22.8	4.5	18.7	4.6

TABLE II
CORPUS ALLOCATION FOR THE TABA CORPUS. TABA IS A GERMAN IN-HOUSE CORPUS COVERING THE DOMAIN OF A TRAIN TIMETABLE INFORMATION SYSTEM FOR MORE THAN 1000 GERMAN TERMINALS. THE CORPUS WAS KINDLY PROVIDED BY PHILIPS AND SCANSOFT

corpus TABA-Speech	training	development	evaluation
amount of acoustic data [h]	13:09	1:41	1:44
silence portion [%]	46.6	46.7	48.0
# speakers	2 585	324	324
# sentences	17 771	2 247	2 390
# running words	58 098	7 502	7 709
# running phonemes	254 362	33 293	33 659
covered vocabulary	1 678	670	669
# lexicon entries	2 972		
# pronunciations	2 988		
# unknown words	–	0	0
perplexity (trigram LM)	7.7	18.0	16.3
perplexity (class-trigram LM)	11.7	17.9	16.1

corpus TABA-NLU	training		development		evaluation	
	src	trg	src	trg	src	trg
# sentences	17 771		2 247		2 390	
# running words	58 098	31 751	7 502	4 083	7 709	4 254
covered vocabulary	1 678	27	670	26	669	26
# singletons	518	0	–	–	–	–
# unknown words	–	–	183	0	169	0
perplexity (trigram LM)	7.7	4.4	18.0	4.8	16.3	4.6

TABLE III
RECOGNITION AND CONFIDENCE MEASURE RESULTS IN TERMS OF WORD ERROR RATE (WER), GRAPH ERROR RATE (GER), AND CONFIDENCE ERROR RATE (CER) FOR THE TELDIR AND TABA CORPUS WITHOUT AND WITH ACOUSTIC ADAPTATION

corpus	TELDIR-Speech development	WER [%]	GER [%]	CER [%] baseline	CER [%]
baseline		13.7	6.2	11.9	10.3
with incr. F-MLLR		12.4	6.2	10.9	8.9

corpus	TELDIR-Speech evaluation	WER [%]	GER [%]	CER [%] baseline	CER [%]
baseline		14.9	8.3	12.5	10.4
with incr. F-MLLR		13.6	7.5	11.2	9.4

corpus	TABA-Speech development	WER [%]	GER [%]	CER [%] baseline	CER [%]
baseline		13.0	7.4	11.2	9.2
with incr. F-MLLR		12.3	7.1	10.6	8.9

corpus	TABA-Speech evaluation	WER [%]	GER [%]	CER [%] baseline	CER [%]
baseline		12.8	7.6	11.0	9.2
with incr. F-MLLR		12.5	7.4	10.7	9.0

substitution errors not detected by the confidence measure, normalized by the total number of recognized words. The CER-baseline is the error rate obtained if no ASR confidence measure is used. The *graph error rate* (GER) is the word error rate of the oracle-best sentences found in the word graphs.

A. Evaluation Metrics

To evaluate the quality of the different approaches, we use the following error criteria.

- **Slot Error Rate (a.k.a. Concept Error Rate)**
  The *slot error rate* (Slot-ER) is similar defined to the well known word error rate and is the ratio of the sum of deleted, inserted, and substituted concepts (slots), normalized by the total number of reference concepts.
- **Attribute Error Rate**
  The *attribute error rate* (AER) is defined as the number of falsely assigned or missed attributes normalized by the total number of attributes. Thus, the AER measures to what extent an approach is able to extract the correct values from the input sentences.
- **Word-based Slot Error Rate**
  The *word-based slot error rate* (word-based Slot-ER) is defined as the number of words that have been aligned to wrong concepts, normalized by the number of all words. Thus, the word-based Slot-ER quantifies the number of wrongly aligned words which cannot be accomplished by the AER. Note that this measure requires reference alignments.

B. Preprocessing

Each source sentence is preprocessed before the sequence of concepts is determined. For both NLU approaches the preprocessing consists of the following steps.

1) **Removal of no-word entities**
   This step mainly affects transcriptions derived from the ASR module. In this step, all no-word entities like silence, hesitations, and other noise occurrences are removed.
2) **Normalization of pronunciation variants**
   Similar to the first step, this step only affects transcriptions obtained from the ASR module. Pronunciation variants are mapped onto the first pronunciation variant which is determined by the order the words occur in the pronunciation lexicon.
3) **Categorization**
   To reduce the number of unseen events, proper names, numbers, and date expressions are mapped onto categories. Categories for proper names can easily be obtained from the entries of the database a dialogue system is connected with.

C. Postprocessing

No postprocessing steps were applied for the ME approach. However, an additional filler-sequence penalty was introduced for decoding that avoids chains of filler concepts. This filler-sequence penalty is similar to the word penalty in ASR. For the source-channel approach, special postprocessing steps were applied for unknown words. Unknown words are words which do not have an entry in the phrase table and are translated by their identity. The steps are applied in the following order.

1) Unknown words that occur at the beginning of a translated sentence are always mapped onto the filler concept.
2) Unknown words that occur after the first position in the target sentence are assigned to their preceding concept.
3) Sequences of equal concepts are reduced to a single concept.

D. Comparison of Alignment Templates and Maximum Entropy Approach Using Text Input

We first determine the alignments between source and target sentences for the TELDIR training corpus using $1^3H^33^44^35^1$ as sequence of models for the source-channel approach (i.e., we apply three iterations of model 1, continue with three iterations of the HMM model, etc.) and $1^4H^43^44^45^4$ for the ME approach. For the TABA corpus, the sequence of model iterations used is $1^4H^43^44^45^4$ for both the source-channel and the ME approach. The optimal training sequences were determined with respect to the error rates obtained on the development sets.

1) Feature Selection, Count Cutoffs, and Smoothing: Feature selection for the feature functions described in the previous section is done on the development corpus of each task. The selection is always done for complete feature types rather than for individual features. For the lexical features, the count cutoff is set to 1, that is, we keep all triples (w, d, c) observed during training. For prefix and suffix features, we additionally require that a word must consist of at least ten characters to generate a feature. The smoothing parameter σ is set to 3 for all corpora.

2) Preprocessing Using Categorizations: A problem that often occurs in NLU tasks is that many semantically relevant words are not covered by the training corpus. For example, it is unlikely that all proper names are observed during training. To improve the results we use categorizations for both approaches and map words like proper names and numbers onto categories. Such categories can easily be obtained from the entries of the database a dialogue system is connected with. Table IV shows an excerpt of the categories used in the TELDIR and

TABLE IV
EXCERPT OF WORD CATEGORIES. IN A PREPROCESSING STEP SOURCE WORDS COVERED BY THE APPLICATION DATABASE ARE MAPPED ONTO CORRESPONDING WORD CATEGORIES

Category	Examples	Category	Examples
$CITY	• Aachen • Köln	$MONTH	• Januar • Februar
$DAYTIME	• Morgen • Vormittag	$CARDINAL	• erster • zweiter
$WEEKDAY	• Montag • Dienstag	$NUMBER	• null • eins
$SURNAME	• Schlegel • Wagner	$FUZZY	• circa • gegen

TABLE V
EFFECT OF DIFFERENT ALIGNMENT MODELS ON THE SLOT ERROR RATE FOR THE TELDIR CORPUS. ERROR RATES ARE PRESENTED WITH AND WITHOUT CATEGORIZATIONS FOR THE ALIGNMENT TEMPLATES (AT) APPROACH AND THE MAXIMUM ENTROPY (ME) FRAMEWORK

corpus TELDIR development	AT		ME	
	Slot-ER [%]	Slot-ER [%] + Categ.	Slot-ER [%]	Slot-ER [%] + Categ.
Model 1	41.4	26.2	23.4	21.6
HMM	21.4	10.9	9.9	4.5
Model 3	20.4	7.4	7.1	3.4
Model 4	19.9	6.4	7.0	2.6
Model 5	19.6	5.8	6.9	**2.5**

corpus TELDIR evaluation	AT		ME	
	Slot-ER [%]	Slot-ER [%] + Categ.	Slot-ER [%]	Slot-ER [%] + Categ.
Model 1	32.9	21.9	24.3	21.2
HMM	15.7	10.7	9.8	4.4
Model 3	14.4	8.1	7.3	4.3
Model 4	13.2	6.5	6.7	3.1
Model 5	13.2	6.3	6.5	**3.1**

TABA corpus. Because the database does not contain all named entities, a large proportion of proper names occurring in the test data remains unobserved.

3) Effect of Alignment Models: Tables V and VII summarize results for both the alignment templates (AT) approach and the ME framework using different alignment models. Both tables present results with and without categorizations. Unlike translations between natural language pairs, word reorderings do not occur in NLU tasks described in this paper. Therefore, we disallow word reorderings for the AT approach by penalizing possible reorderings with additional costs set to infinity. Because the ME approach basically works similar to a tagger, the monotonicity of the word-concept alignment is guaranteed for this method.

The ME results presented in Tables V and VII are based on the six feature types listed in Tables VI and VIII. Starting with only lexical features, we successively extend the model by including additional feature functions. The results show that the ME models clearly outperform the AT approach. The quality of the AT approach is achieved within the ME framework by just including lexical and transition features, and is significantly improved by adding further features.

Another interesting effect when looking at the results obtained without categorizations is that the ME framework suffers less from unknown words compared to the AT approach. The problem can be mitigated but not eliminated if more training data is available as is the case for the TABA corpus. The reason

TABLE VI
DEPENDENCE OF ERROR RATES ON THE NUMBER OF INCLUDED FEATURE TYPES FOR THE TELDIR CORPUS USING ME. THE SOURCE SENTENCES ARE PREPROCESSED USING CATEGORIZATIONS. THE FEATURES WERE TRAINED ON THE MODEL 5 ALIGNED TRAINING CORPUS

corpus TELDIR development	total # features	Slot-ER [%] + Categ.	word-based Slot-ER [%]	AER [%]
lexical	60,287	5.5	5.7	2.7
+ prior	60,330	5.0	5.6	2.6
+ transition	62,179	3.2	5.6	2.3
+ pre- & suffix	91,290	2.7	5.3	2.3
+ compound	130,893	2.5	5.3	2.1
+ capitalization	130,936	**2.5**	5.3	1.8

corpus TELDIR evaluation	total # features	Slot-ER [%]	word-based Slot-ER [%]	AER [%]
lexical	60,287	5.5	6.6	3.4
+ prior	60,330	5.3	6.4	3.3
+ transition	62,179	3.5	6.6	2.2
+ pre- & suffix	91,290	3.3	6.4	1.6
+ compound	130,893	3.1	6.3	1.7
+ capitalization	130,936	**3.1**	6.4	1.6

TABLE VII
EFFECT OF DIFFERENT ALIGNMENT MODELS ON THE SLOT ERROR RATE FOR THE TABA CORPUS. ERROR RATES ARE PRESENTED WITH AND WITHOUT CATEGORIZATIONS FOR THE ALIGNMENT TEMPLATES (AT) APPROACH AND THE MAXIMUM ENTROPY (ME) FRAMEWORK

corpus TABA development	AT		ME	
	Slot-ER [%]	Slot-ER [%] + Categ.	Slot-ER [%]	Slot-ER [%] + Categ.
Model 1	13.6	8.8	12.0	12.4
HMM	13.3	8.7	6.6	4.0
Model 3	11.3	7.8	6.4	2.9
Model 4	10.8	6.0	6.2	2.6
Model 5	10.8	5.2	6.2	**2.5**

corpus TABA evaluation	AT		ME	
	Slot-ER [%]	Slot-ER [%] + Categ.	Slot-ER [%]	Slot-ER [%] + Categ.
Model 1	13.0	8.3	11.6	11.5
HMM	12.3	7.4	5.7	3.4
Model 3	11.3	6.9	5.3	3.1
Model 4	11.0	5.5	4.9	2.7
Model 5	11.1	5.2	4.8	**2.6**

is that the ME model can use variable-length contexts flexibly. Whereas the template approach either has an appropriate template or not and each word can occur in at most one template, the ME model can reuse contexts flexibly for various adjacent words.

Comparing the performance on both the TELDIR and the TABA task, we see that the error rates are much lower for the TABA task than for the TELDIR task; the reason is due to the very limited training data of the TELDIR task. However, we still achieve good results using the ME framework, even if the training corpus is very small.

4) Effect of Context Lengths: Both the AT approach as well as the feature functions defined for the ME framework make use of local word contexts, that is, predecessor and successor words are explicitly taken into account. For the AT approach, we have chosen a maximum template length of 7 for both corpora. This value was selected empirically from the TABA corpus in Fig. 7.

As can be derived from the figure, a larger value does not improve the Slot-ER. The feature functions for the ME model use a shorter context of five consecutive words. Augmenting the

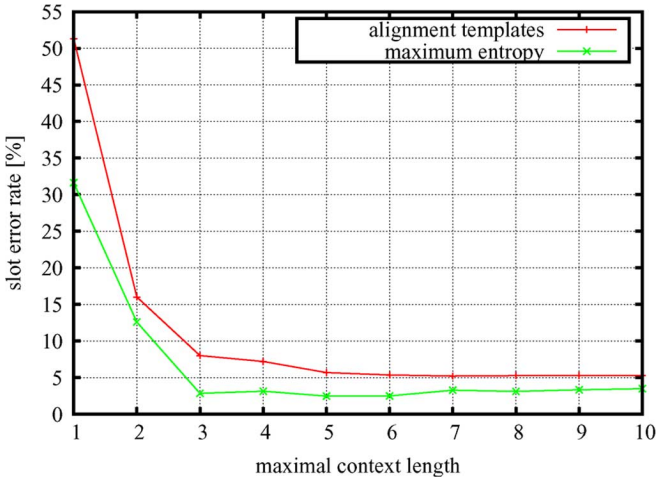

Fig. 7. Effect of the maximal allowed context length on the slot error rate for both the AT approach and the ME-based framework where a symmetric context window is used. The plot shows error rates for the TABA development corpus using categorizations.

TABLE VIII
DEPENDENCE OF ERROR RATES ON THE NUMBER OF INCLUDED FEATURE TYPES FOR THE TABA CORPUS USING ME. THE SOURCE SENTENCES ARE PREPROCESSED USING CATEGORIZATIONS. THE FEATURES WERE TRAINED ON THE MODEL 5 ALIGNED TRAINING CORPUS

corpus TABA development	total # features	Slot-ER [%] + Categ.	word-based Slot-ER [%]	AER [%]
lexical	226,152	7.5	5.1	6.7
+ prior	226,205	5.8	4.0	6.1
+ transition	229,014	3.0	3.4	3.6
+ pre- & suffix	229,226	3.0	3.4	3.6
+ compound	827,861	2.5	2.6	3.3
+ capitalization	827,914	**2.5**	2.6	3.3

corpus TABA evaluation	total # features	Slot-ER [%]	word-based Slot-ER [%]	AER [%]
lexical	226,152	6.8	4.8	6.4
+ prior	226,205	4.8	3.6	5.8
+ transition	229,014	2.8	3.1	4.0
+ pre- & suffix	229,226	2.8	3.1	4.0
+ compound	827,861	2.7	2.7	3.9
+ capitalization	827,914	**2.6**	2.7	3.9

context length slightly deteriorates the performance and results in an overfitted model.

5) Effect of Amount of Training Data on Error Rate: The amount of training data used for the TELDIR and the TABA corpus differs roughly by a factor of 40. Since the source sentences need to be annotated manually by sequences of formal concepts in order to apply supervised learning techniques, the question is what degradation in error rate we have to expect if we reduce the amount of training data used. This is important if an NLU component is to be trained for a new domain where only little data is available. Because additional training data should be collected from dialogue transactions of real users interacting with a fully functional and running dialogue system, a curve plotting the NLU performance against the amount of training data gives some hints on how much training data should be used for setting up a new system. Fig. 8 shows the performance of the NLU component in the course of the amount of training data used. Both plots use the full set of

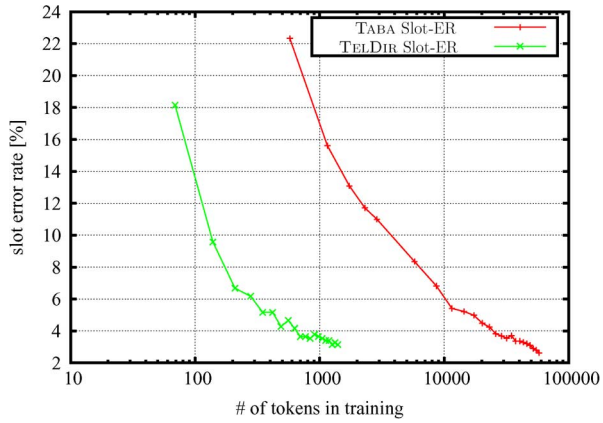

Fig. 8. Effect of amount of training data on the slot error rate for the TELDIR and TABA evaluation corpus using the ME-based approach.

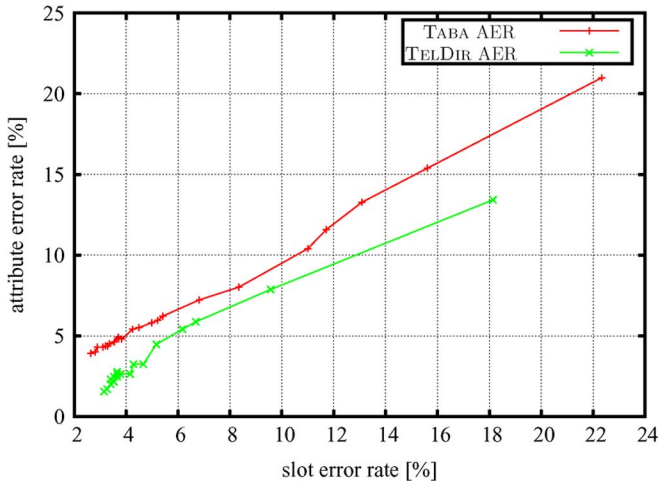

Fig. 9. Attribute error rate in course of slot error rate for the TELDIR and TABA evaluation corpus using the ME-based approach.

feature functions described in Section IV-A. In order to get a reasonable Slot-ER, it is sufficient to have around 1000 training tokens for the TELDIR task. The TABA task is more complex due to its time and date expressions. Here, a Slot-ER of around 6% can be achieved with 10 000 training samples. Although both tasks differ in their complexity, they almost have the same slope.

6) Effect of Slot Error Rate on Attribute Error Rate: Each concept is associated with a (possibly empty) set of attributes. Because a wrongly assigned concept also affects the AER, we want to analyze this effect more thoroughly. Fig. 9 plots the AER in course of the Slot-ER using the ME approach. Increasing Slot-ERs were obtained by successively reducing the number of training data (see Fig. 8). For both tasks, the AER is proportional to the Slot-ER. This effect partly results from the categorizations used which yield a good generalization even if only a few training samples have been used.

E. Comparison of Alignment Templates and Maximum Entropy Approach Using Speech Input

If the input modality is speech, inputs to the NLU component can contain speech recognition errors. Furthermore,

TABLE IX
SPEECH UNDERSTANDING RESULTS FOR THE TELDIR AND TABA CORPUS

corpus TELDIR-SLU development	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
text reference	–	5.8	4.1	2.5	1.8
ASR	13.7	11.3	22.1	7.8	20.4
with incr. F-MLLR	12.4	10.6	20.6	6.8	19.1

corpus TELDIR-SLU evaluation	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
text reference	–	6.3	4.6	3.1	1.6
ASR	14.9	15.2	26.0	11.8	23.0
with incr. F-MLLR	13.6	14.1	22.7	10.6	20.6

corpus TABA-SLU development	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
text reference	–	5.2	7.9	2.5	3.3
ASR	13.0	14.5	18.9	11.8	15.2
with incr. F-MLLR	12.3	14.0	18.5	11.4	14.4

corpus TABA-SLU evaluation	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
text reference	–	5.2	6.9	2.6	3.9
ASR	12.8	13.6	16.1	11.6	13.6
with incr. F-MLLR	12.5	13.4	16.0	11.6	13.4

spontaneous speech effects like disfluencies, hesitations, and stressed speech can deteriorate the performance of the ASR module and, thus, the quality of the NLU component. Therefore, we want to investigate how both approaches perform if the input modality is speech. For this purpose, both models were trained on sentence pairs using reference transcriptions. Table IX summarizes Slot-ERs and AERs for both tasks. For comparison with Section VII-D, the first row contains error rates for text-based results.

If we compare the source-channel approach with the direct model, we see that the ME approach clearly outperforms the AT approach which is in accordance to the results obtained from text input. However, compared to the TABA corpus, the AER for the TELDIR corpus deteriorates much more from erroneous input data. Apart from less training data, this effect results from erroneous recognitions of spelling sequences. The TELDIR domain allows users to spell names which is modeled by a spelling concept. The attribute of a spelling concept is the sequence of letters uttered by a user. If this sequence contains only one wrong letter, the concept remains most likely a spelling concept, but the attribute value is considered to be wrong, independent of the total number of letters that were recognized correctly. To mitigate this effect, we can use the proper names contained in the application database as an additional knowledge source. The idea is to extract all proper names from the database and decompose them into character sequences. Doing this we can match a recognized spelling sequence with the character sequence obtained from proper names contained in the database. If a sufficiently number of characters matches a decomposed database entry, we assume that the remaining unmatched characters resulted from an erroneous recognition and replace the sequence by the best matching database entry. The number of sufficiently matching characters must be chosen such

TABLE X
SPEECH UNDERSTANDING RESULTS FOR THE TELDIR CORPUS USING AUTOMATIC SPELLING CORRECTION

corpus TELDIR-SLU development	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
baseline	12.4	10.6	20.6	6.8	19.1
+ spelling correction	12.4	10.6	19.1	6.8	17.3

corpus TELDIR-SLU evaluation	WER [%]	AT		ME	
		Slot-ER [%] + Categ.	AER [%]	Slot-ER [%] + Categ.	AER [%]
baseline	13.6	14.1	22.7	10.6	20.6
+ spelling correction	13.6	14.1	20.7	10.6	18.4

that on the one hand we benefit from using the database as additional knowledge source while on the other hand we allow for accepting new spelling sequences that are not contained in the database. The threshold for the mutual error rate between a hypothesized spelling sequence and a database entry was optimized on the development set and is set to $\frac{1}{3}$. Results are listed in Table X.

F. Combining Speech Recognition and Language Understanding

To combine ASR with NLU we proceed as follows: we first generate ASR word graphs for all utterances and compute confidence values for each hypothesis contained in a word graph. We then extract N-best lists from the word graphs and process each N-best entry with the ME approach, thus enriching the N-best lists with the additional features described in Section V-C. The enriched N-best lists are then inserted into a repository on which the minimum error rate training algorithm is applied. A problem with this procedure is that the ASR decoder cannot produce better results in subsequent iterations because the new features are not integrated into the ASR decoding process, but are added after the N-best lists have been extracted. Since the decoder cannot use the new knowledge sources directly, it cannot produce new entries for the repository. Therefore, we have to make sure that word graphs and N-best lists contain a sufficiently large number of hypotheses after the first iteration. According to Table XI, a 10×10-best list provides enough sentence alternatives from which we can select promising candidates in order to lower the error rate. Furthermore, selecting a candidate out of a small N-best list might be more realistic than striving to find the oracle-best candidate in a huge N-best list. The motivation behind this is that the additional features should rather aim at reranking candidates whose scores are very close to each other, than trying to move a very low-ranked candidate to the first position. The final repository contains a 10×10 best list for each utterance where each N-best entry consists of three streams together with the associated error counts: the word stream obtained from ASR decoding, the concept stream obtained from NLU decoding, and the attribute stream obtained from the attribute extraction. The N-best lists do not contain duplicates. The hypothesized word-concept segmentation is kept for the concept stream in order to simplify the attribute extraction.

1) Minimizing the Word Error Rate: We first investigate whether NLU features can help reducing ASR errors. Table XII

TABLE XI
ORACLE ERROR RATES FOR THE SPOKEN LANGUAGE UNDERSTANDING COMPONENT. CONCEPT GRAPHS ARE PRODUCED BY APPLYING THE ME APPROACH ON THE FIRST BEST RESULTS FROM THE ASR COMPONENT. THE 100-BEST LISTS ARE DIRECTLY EXTRACTED FROM THE CONCEPT GRAPHS WHEREAS THE 100-BEST* LISTS ARE OBTAINED BY MERGING 10-BEST CONCEPT LISTS DERIVED FROM 10 CONCEPT DECODINGS. EACH DECODING USES ONE CANDIDATE FROM A 10-BEST ASR LIST AS INPUT

corpus TELDIR-SLU	baseline Slot-ER [%]	graph Slot-ER [%]	100-best Slot-ER [%]	100-best* Slot-ER [%]
development	6.8	1.9	2.4	1.7
evaluation	10.6	2.6	2.8	2.3

corpus TABA-SLU	baseline Slot-ER [%]	graph Slot-ER [%]	100-best Slot-ER [%]	100-best* Slot-ER [%]
development	11.4	2.5	3.9	3.1
evaluation	11.6	2.7	4.0	3.5

TABLE XII
COMBINATION OF SPEECH RECOGNITION AND NATURAL LANGUAGE UNDERSTANDING FOR THE TELDIR AND TABA TASK THE OBJECTIVE FUNCTION USED IS MINIMIZING THE WORD ERROR RATE

corpus TELDIR-Speech Features	development WER [%]	evaluation WER [%]
Baseline (acu + lm + word-penalty)	12.4	13.6
Baseline (optimized on Repository)	12.2	13.8
+ concept score	11.7	12.9
+ spellingCorrection	10.9	12.3
+ concept penalty	**10.7**	**12.4**
+ asr posterior	10.9	12.3
+ asr confidence	10.9	12.3
+ concept LM	11.0	13.1

corpus TABA-Speech Features	development WER [%]	evaluation WER [%]
Baseline (acu + lm + word penalty)	12.3	12.5
Baseline (optimized on Repository)	11.8	11.7
+ concept score	11.7	11.4
+ concept penalty	11.7	11.4
+ concept LM	**11.6**	**11.5**

presents results for combining ASR with NLU. The feature weights were optimized on the development set according to the algorithm described in Section V-B. The objective function used is minimizing the WER. The first line in each table is the baseline WER obtained under realtime recognition constraints which is equal to the ASR performance reported in Table III. To measure the effect of newly added features, we first optimize the weights for the ASR baseline features on the development N-best repository and apply those weights on the evaluation N-best repository. The ASR baseline features comprise the acoustic model, the language model, and the word penalty. For the TELDIR task this gives a small improvement of 0.2% on the development set which does not generalize to the evaluation set. For the TABA task, the baseline optimization reduces the WER much more on both the development and the evaluation corpus. Due to real-time constraints we cannot use the optimized feature weights directly during speech decoding since this would require larger beam settings. Therefore, applying the optimized baseline feature weights on the N-best repository is similar to a second-pass decoding, where we rescore the candidates from the first pass with optimized weights.

The first added feature is the NLU concept score. With this feature, the WER is reduced by 0.6% on the TELDIR development set and by 0.5% on the evaluation set. The improvements on the TABA task are smaller but consistent. A special feature that we can use for the TELDIR corpus is the spelling correction feature. We already showed in Section VII-E that this feature reduces the AER by comparing the actual attribute values with the values stored in the application database. We can use the same knowledge source here and use the Levenshtein distance of a hypothesized spelling sequence w.r.t. the closest database entry as additional feature. Because only a fraction of the spelled proper names is covered by the database, we leave it to the minimum error rate training framework to find a tradeoff between covered and uncovered proper names.

TABLE XIII
COMBINATION OF SPEECH RECOGNITION AND NATURAL LANGUAGE UNDERSTANDING FOR THE TELDIR AND TABA TASK THE OBJECTIVE FUNCTION USED IS MINIMIZING THE SLOT ERROR RATE

corpus TELDIR-Speech Features	development Slot-ER [%]	evaluation Slot-ER [%]
Baseline (conceptScore)	6.8	10.6
Baseline (1st best on Repository)	9.9	13.9
+ acoustic model	8.3	11.7
+ language model	6.3	10.8
+ word penalty	**6.3**	**10.1**

corpus TABA-Speech Features	development Slot-ER [%]	evaluation Slot-ER [%]
Baseline (conceptScore)	11.4	11.6
Baseline (1st best on Repository)	17.1	17.7
+ acoustic model	14.9	15.6
+ language model	11.0	11.3
+ word penalty	**10.6**	**10.7**

2) Minimizing the Slot Error Rate: Now, we investigate if we can reduce the Slot-ER by adding ASR-related features to the NLU features. Table XIII presents results for combining ASR with NLU when minimizing the Slot-ER. The baseline Slot-ER is equal to the Slot-ER reported in Table XI. Note that this is a single feature that employs all the feature functions described in Section IV-A. Therefore, no baseline optimization on the N-best repository is necessary. However, the first-best result determined on the repository might actually be worse compared to the baseline result given that we use only the concept score as a single feature. The reason is that the baseline result is computed from the first best ASR sentence hypothesis for which the ASR features like the acoustic model weight, the language model weight, etc., are already fixed. If we take only the concept score into account and rescore the repository, we decouple the concept feature function from the ASR knowledge sources and allow choosing much shorter sentence hypotheses which might have more deletion errors but produce lower concept scores. As soon as we add more ASR feature functions, we observe a reduction in terms of Slot-ERs. Optimizing the weights for all standard ASR feature functions, that is, adding acoustic scores, language model scores, and word penalties, significantly reduces the Slot-ER.

3) Minimizing Word and Slot Error Rate Simultaneously: By combining the error counts for ASR and NLU and by using all features defined previously, we can minimize both WER and Slot-ER simultaneously. The unweighted error counts are combined by summation. Results are presented in Table XIV.

TABLE XIV
COMBINATION OF SPEECH RECOGNITION AND NATURAL LANGUAGE UNDERSTANDING. THE OBJECTIVE FUNCTION USED IS MINIMIZING THE SLOT AND THE WORD ERROR RATE SIMULTANEOUSLY

corpus TELDIR-Speech	development		evaluation	
Features	WER [%]	Slot-ER [%]	WER [%]	Slot-ER [%]
Baseline	12.4	6.8	13.6	10.6
+ all	10.9	6.7	12.3	9.6

corpus TABA-Speech	development		evaluation	
Features	WER [%]	Slot-ER [%]	WER [%]	Slot-ER [%]
Baseline	12.3	11.4	12.5	11.6
+ all	11.6	10.3	11.4	10.2

VIII. SUMMARY AND CONTRIBUTIONS

In this paper, we proposed two approaches to SLU based on statistical machine translation: a source channel-based approach and a ME-based approach. We evaluated both approaches on two NLU tasks which were derived from different domains and showed that the ME approach clearly outperforms the source channel-based method within these settings. In contrast to [19], where almost no context was taken into account, we showed that local contexts are crucial to this task. The ME approach described in [10] was evaluated only on pure text data and required a separate decoder that supplies a list of candidate translations on which then the ME model was trained. Here, we directly trained the ME model on bilingual training data. We analyzed the quality of different alignment models and feature functions and investigated several effects, including the effect of categorizations, different context lengths, and the relation between Slot-ERs and AERs. We also analyzed the performance of both approaches when speech was used as input modality. Finally, we showed how ASR and NLU-based knowledge sources can be log-linearly combined such that a combination of error rates is minimized. This provided a tighter coupling between ASR and NLU. We employed the minimum error rate training framework in order to determine an optimal set of feature weights and showed that the WER could be reduced using NLU-based feature functions, and that, correspondingly, the Slot-ER was reduced using ASR-based feature functions. In contrast to [24], we used a different optimization technique which computes the error surface for each feature function used. Furthermore, we investigated a larger set of feature functions and took sentence-based features into account.

ACKNOWLEDGMENT

The authors would like to thank Philips Speech Processing and Scansoft for kindly providing the TABA corpus.

REFERENCES

[1] Oerder and H. Ney, "Word graphs: An efficient interface between continuous-speech recognition and language understanding," in *Proc. ICASSP*, Minneapolis, MN, Apr. 1993, vol. 2, pp. 119–122.

[2] Y.-Y. Wang and A. Acero, "SGStudioL rapid semantic grammar development for spoken language understanding," in *Proc. Interspeech*, Lisbon, Portugal, Sep. 2005, vol. 2, pp. 1869–1872.

[3] E. Levin and R. Pieraccini, "Concept-based spontaneous speech understanding system," in *Proc. Eurospeech*, Madrid, Spain, Sep. 1995, vol. 2, pp. 555–558.

[4] S. Miller, R. Bobrow, R. Schwartz, and R. Ingria, "Statistical language processing using hidden understanding models," in *Proc. Human Lang. Technol. Workshop Sponsored by ARPA*, Plainsboro, NJ, Mar. 1994, pp. 278–282.

[5] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, "The mathematics of machine translation: Parameter estimation," *Comput. Lingist.*, vol. 19, no. 2, pp. 263–311, 1993.

[6] S. Vogel, H. Ney, and C. Tillmann, "HMM-based word alignment in statistical translation," in *Proc. COLING*, Copenhagen, Denmark, Aug. 1996, vol. 2, pp. 836–841.

[7] F. J. Och, C. Tillmann, and H. Ney, "Improved alignment models for statistical machine translation," in *Proc. EMNLP/VLC*, College Park, MD, Jun. 1999, pp. 20–28.

[8] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Comput. Lingist.*, vol. 29, no. 1, pp. 19–51, 2003.

[9] A. Berger, "Improved iterative scaling: A gentle introduction," 1997 [Online]. Available: http://www.cs.cmu.edu/afs/cs/user/aberger/www/ps/scaling.ps

[10] K. A. Papineni, S. Roukos, and R. T. Ward, "Feature-based language understanding," in *Proc. Eurospeech*, Rhodes, Greece, Sept. 1997, pp. 1435–1438.

[11] E. T. K. Sang and S. Buchholz, "Introduction to the CoNLL-2000 shared task: Chunking," in *Proc. CoNLL-LLL*, Lisbon, Portugal, Sep. 2000, pp. 127–132.

[12] O. Bender, K. Macherey, F.-J. Och, and H. Ney, "Comparison of alignment templates and maximum entropy models for natural language understanding," in *Proc. EACL*, Budapest, Hungary, Apr. 2003, pp. 11–18.

[13] A. McCallum, D. Freitag, and F. Pereira, "Maximum entropy markov models for information extraction and segmentation," in *Proc. ICML*, Stanford, CA, Jun. 2000, pp. 591–598.

[14] S. F. Chen and R. Rosenfeld, A Gaussian prior for smoothing maximum entropy models School of Comput. Sci. Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-99-108, Feb. 1999.

[15] P. Beyerlein, "Diskriminative modellkombination in spracherkennung mit gro em wortschatz," Ph.D. dissertation, Comput. Sci. Dept., RWTH Aachen Univ., Aachen, Germany, 2000.

[16] F. J. Och, "Minimum error rate training in statistical machine translation," in *Proc. ACL*, Sapporo, Japan, Jul. 2003, pp. 160–167.

[17] J. L. Bentley and T. A. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Trans. Comput.*, vol. C-28, no. 9, pp. 643–647, Sep. 1979.

[18] F. Wessel, K. Macherey, and R. Schlüter, "Using word probabilities as confidence measures," in *Proc. ICASSP*, Seattle, WA, May 1998, vol. 1, pp. 225–228.

[19] M. Epstein, K. Papineni, S. Roukos, T. Ward, and S. Della Pietra, "Statistical natural language understanding using hidden clumpings," in *Proc. ICASSP*, Atlanta, GA, May 1996, vol. 1, pp. 176–179.

[20] S. Miller, M. Bates, R. Ingria, J. Makhoul, and R. Schwartz, "Recent progress in hidden understanding models," in *Proc. ARPA Spoken Lang. Technol. Workshop*, Austin, TX, Jan. 1995, pp. 276–280.

[21] Y. He and S. Young, "A data-driven SLU system," in *Proc. ASRU*, St. Thomas, U.S. Virgin Islands, Dec. 2003, pp. 583–588.

[22] P. Haffner, G. Tur, and J. Wright, "Optimizing SVMs for complex call classification," in *Proc. ICASSP*, Hong Kong, China, Apr. 2003, vol. 1, pp. 632–635.

[23] N. Gupta, G. Tu, and D. Hakkani-Tür, "The AT&T SLU system," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 1, pp. 213–222, Jan. 2006.

[24] E. Matusov, S. Kanthak, and H. Ney, "On the integration of speech recognition and statistical machine translation," in *Proc. Interspeech*, Lisbon, Portugal, Sep. 2005, pp. 3177–3180.

[25] K. Sudoh and H. Tsukada, "Tightly integrated spoken language understanding using word-to-concept translation," in *Proc. Interspeech*, Lisbon, Portugal, Sep. 2005, vol. 1, pp. 429–432.

[26] T. J. Hazen, T. Burianek, J. Polifroni, and S. Seneff, "Integrating recognition confidence scoring with language understanding and dialogue modeling," in *Proc. ICSLP*, Beijing, China, Oct. 2000, vol. 2, pp. 1042–1045.

[27] G. Tur, J. Wright, A. Gorin, G. Riccardi, and D. Hakkani-Tur, "Improving spoken language understanding using word confusion networks," in *Proc. ICSLP*, Denver, CO, Sep. 2002, pp. 1137–1140.

[28] H. Aust, M. Oerder, F. Seide, and V. Steinbiss, "The philips automatic train timetable information system," *Speech Commun.*, vol. 17, pp. 249–262, Nov. 1995.

Klaus Macherey received the Diploma degree in computer science from RWTH Aachen University, Aachen, Germany. in 1999.

From May 1999 until March 2006, he was a Research Assistant in the Computer Science Department 6, RWTH Aachen. Since the end of March 2006, he has been a Research Scientist at Google, Inc. Mountain View, CA. His primary research interests cover machine translation, speech recognition, natural language understanding, dialogue systems, and reenforcement learning.

Oliver Bender received the Diploma degree in computer science from RWTH Aachen University, Aachen, Germany, in 2002.

Since then, he has been a Research Assistant with the Computer Science Department 6, RWTH. His primary research interests cover statistical machine translation, natural language understanding, spoken dialogue systems, and information extraction. In November 2008, he joined Nuance Communications Aachen GmbH.

Hermann Ney (M'86–SM'07) received the Diploma degree in physics from Goettingen University, Goettingen, Germany, in 1977 and the Dr.-Ing. degree in electrical engineering from Braunschweig University of Technology, Braunschweig, Germany, in 1982.

He is a Full Professor of computer science at RWTH Aachen University, Aachen, Germany. In 1977, he joined Philips Research in Germany, and since 1985 he has headed the Speech Recognition Group that pioneered the first prototype systems for large vocabulary continuous speech recognition and spoken dialogue systems. From October 1988 to October 1989, he was a Visiting Scientist at Bell Laboratories, Murray Hill, NJ. In July 1993, he joined the Computer Science Department, RWTH Aachen University. His responsibilities include planning, directing, and carrying out research for national, European, and industrial sponsors and supervising Ph.D. students. His main research interests lie in the area of statistical methods for pattern recognition and human language technology and their specific applications to speech recognition, machine translation, and image object recognition. In particular, he has worked on dynamic programming and discriminative training for speech recognition, on language modeling and on phrase-based approaches to machine translation. His work has resulted in more than 400 conference and journal papers.

Prof. Ney was the recipient of the Technical Achievement Award of the IEEE Signal Processing Society in 2005.