Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Speech and Language 20 (2006) 495–514

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

Beyond ASR 1-best: Using word confusion networks in spoken language understanding

Dilek Hakkani-Tür [a,*], Frédéric Béchet [b], Giuseppe Riccardi [c,1], Gokhan Tur [a]

[a] *AT&T Labs – Research, Florham Park, NJ 07932, United States*
[b] *LIA-CNRS, University of Avignon, BP1228, 84911 Avignon Cedex 09, France*
[c] *University of Trento, 12 38100, Italy*

Received 23 January 2004; accepted 6 July 2005
Available online 10 October 2005

Abstract

We are interested in the problem of robust understanding from noisy spontaneous speech input. With the advances in automated speech recognition (ASR), there has been increasing interest in spoken language understanding (SLU). A challenge in large vocabulary spoken language understanding is robustness to ASR errors. State of the art spoken language understanding relies on the best ASR hypotheses (ASR 1-best). In this paper, we propose methods for a tighter integration of ASR and SLU using word confusion networks (WCNs). WCNs obtained from ASR word graphs (lattices) provide a compact representation of multiple aligned ASR hypotheses along with word confidence scores, without compromising recognition accuracy. We present our work on exploiting WCNs instead of simply using ASR one-best hypotheses. In this work, we focus on the tasks of named entity detection and extraction and call classification in a spoken dialog system, although the idea is more general and applicable to other spoken language processing tasks. For named entity detection, we have improved the F-measure by using both word lattices and WCNs, 6–10% absolute. The processing of WCNs was 25 times faster than lattices, which is very important for real-life applications. For call classification, we have shown between 5% and 10% relative reduction in error rate using WCNs compared to ASR 1-best output.
© 2005 Elsevier Ltd. All rights reserved.

1. Introduction

User-initiative spoken dialog systems enable humans to express their needs in spoken natural language, in contrast to machine-initiative systems. Such systems automatically extract the meaning from speech input and act upon what people say. Two key components in goal driven human–machine conversational systems are utterance intent determination (call classification) and corresponding argument extraction (named entity extraction) (Gupta et al., To appear). For example, if the user says "I would like to get my balance for the account number

[*] Corresponding author.
E-mail addresses: dtur@research.att.com (D. Hakkani-Tür), frederic.bechet@lia.univ-avignon.fr (F. Béchet), dsp3@research.att.com (G. Riccardi), gtur@research.att.com (G. Tur).
[1] The work was done while the third author was at the AT&T.

0885-2308/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.csl.2005.07.005

1 2 3 4 5 6", then the corresponding intent or semantic label (call-type) would be "Request(Balance)" and an argument or parameter for this call-type, i.e., the account number, would be "123456."

Typically spoken language processing systems are composed of sequential and independent components, starting from an automatic speech recognizer (ASR). Further components, such as spoken language understanding (SLU) use the best hypothesis output of ASR (ASR 1-best) (Gorin et al., 2002; Wang et al., 2003). On the other hand, in such systems, it is very important to be robust to ASR errors. Especially with spontaneous telephone speech, the typical word error rate (WER) for ASR 1-best output is around 25–30%; in other words, one in every three–four words is misrecognized (Kingsbury et al., 2003; Riccardi and Hakkani-Tür, 2003). Misrecognizing a word may result in misunderstanding the whole utterance, even though all other words are correct. For example, in a named entity extraction system, assuming that an account number should include 6 consecutive digits, misrecognition of any of the digits as a non-number will prevent the correct extraction of this named entity.

More robust systems use ASR *n*-best hypotheses instead of just using the best one (Stolcke et al., 1997; Goel et al., 1998; He and Young, 2003). Another solution is to compute ASR word confidence scores (Wessel et al., 1999; Cox and Dasmahapatra, 2002; Hakkani-Tür and Riccardi, 2003; among others) and use them during understanding tasks to prevent errors caused by misrecognized words (Rose et al., 2001; Tur et al., 2002; Hazen et al., 2002; Cox and Cawley, 2003).

Word lattices are used to approximate the word search space in large vocabulary continuous speech recognition (LVCSR) systems. Usually, they are acyclic and have no a priori structures. Their transitions are weighted by the acoustic and language model probabilities. Using lattices as input to an SLU system is very promising, because the oracle accuracy of lattices is much higher than the word accuracy of ASR 1-best hypotheses (Oerder and Ney, 1993; Tur et al., 2002). The oracle accuracy is the accuracy of the path in a lattice closest to the reference transcriptions. An extra benefit for SLU is that lattices include multiple hypotheses of the recognized utterance, hence can be useful in tolerating some amount of syntactic variability.

In the literature, Johnson et al. (1998) suggested using word graphs for interfacing acoustic speech recognition systems with natural language processing systems. Inspired by our previous work (Tur et al., 2002), Cortes et al. (2003) have introduced lattice kernels for call classification.

More recently, a new class of *normalized* word lattices have been proposed (Mangu et al., 2000; Hakkani-Tür and Riccardi, 2003). These word lattices (a.k.a. word confusion networks, (WCNs)) are more efficient than traditional word lattices, in terms of size and structure, without compromising recognition accuracy. They also provide an alignment for all the strings in the word lattices. The general structure of these lattices and WCNs are shown in Fig. 1. Since WCNs force the competing words to be in the same group, they enforce the alignment of the words that occur at the same approximate time interval in the lattice. This time alignment

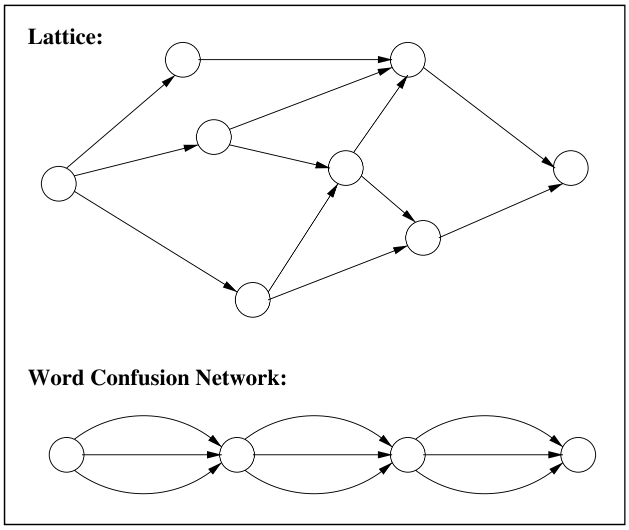

Fig. 1. Typical structures of lattices and WCNs.

may be very useful in language processing. The words in the WCNs have posterior probabilities, which can be used as their confidence scores. These are basically the sum of the probabilities of all paths which contain that word at around that approximate time frame. We use the confidence scores of the words in the WCNs during understanding to achieve robustness to ASR errors. WCNs are much smaller than ASR lattices and they still have comparable word accuracy using their best path and even better oracle accuracy.

In this paper, we describe our pivot algorithm to organize the lattice output of ASR into a WCN and show that WCNs are not only useful for decreasing ASR word error rate, but they also outperform traditional word lattices or ASR *n*-best hypotheses for spoken language understanding. There are many applications where the WCNs have been very useful. In our previous work, we have employed WCNs for unsupervised learning of language models (Gretter and Riccardi, 2001), active learning for ASR (Hakkani-Tür et al., 2002), machine translation (Bangalore et al., 2001), call classification (Tur et al., 2002), and named entity extraction (Béchet et al., 2002). This paper presents the general idea of exploiting the WCNs obtained from the lattice output of ASR in spoken dialog systems. In this work, we extend the previous study on call classification (Tur et al., 2002), and show that it is also possible to use WCNs with a discriminative classifier using the *n*-grams extracted from the WCNs along with their confidences. We also show the advantages of using WCNs when compared to lattices, for named entity extraction.

In the following section, we describe the pivot algorithm. We then address how we use the WCNs in understanding, namely in detecting/extracting named entities (Section 3) and classifying the calls into one of the predefined call-types (Section 4).

2. Word confusion networks

In this section, we describe a general algorithm for generating word confidence scores and confusion networks from the ASR lattice output, the pivot algorithm. We describe the algorithm here in the context of automatic speech recognition (ASR). This algorithm has the clear advantage of *normalizing* the search space of word hypotheses. The advantages of these normalized lattices are in terms of memory and computation:

- **Size** WCNs are much smaller in size than word lattices, while preserving accuracy of the original search space.
- **Structure** WCNs have the *compositional property*, they are composed of a sequence of sub-lattices. Suppose that we want to compute the word string with lowest weight, $C_{\min}$, among all strings $W = w_1,\ldots,w_k,\ldots,w_N$ in the lattice, we can divide the problem into sub-problems of the same nature

$$C_{\min} = \min_W \sum_{w_k} c_{i,j}^{w_k} = \sum_{m_l} \min_{W_l} \sum_{v_k} c_{i,j}^{v_k}, \tag{1}$$

where $W_l = v_1,\ldots,v_k,\ldots,v_M$ is the set of word strings recognized by the lth sub-lattice, m_l.

2.1. The pivot algorithm

Word confusion network computation has been introduced by Mangu et al. (2000). In that work, the words (the labels of transitions) are clustered according to the similarity of their pronunciations and occurrence times, and the WCN is formed using the order of these clusters in the ASR output lattice. This algorithm is designed to reduce word error rate for the task of ASR.

The pivot approach (Hakkani-Tür and Riccardi, 2003) used in this paper aims at *normalizing* the topology of an input graph, generating a WCN. The labels on the graph are generic and could be words as well as part of speech tags, parse tags, or phones. The confidence scores are computed directly from the labeled graph. We describe the algorithm here in the context of ASR, so for the purpose of this paper, we assume that the transition labels are words.

The first step of the algorithm is the computation of the posterior probability of each transition in the ASR lattice, by applying the forward and backward algorithms to the word graph. At this point, the posterior probability of a transition could be used as a confidence score by itself, but some improvement is possible by taking into account the competing hypotheses in the same time slot. We define time slot $ts(T) = [t(S_s^T), t(S_e^T)]$ of a transition T as the speech interval between the starting (S_s^T) and ending (S_e^T) states of T, where $t(X)$ denotes the frame number (i.e., state time) corresponding to the state X.

```
1. Compute the posterior probabilities of all transitions $T$
   in the word lattice.
2. Extract the pivot baseline path.
3. For all transitions $T$ in the topologically ordered
   lattice, do:
 (1) Find the two consecutive states $\hat{S}_s$ and $\hat{S}_e$ that have the
     maximum overlap, $O(T, \hat{S}_s, \hat{S}_e)$, with $T$.
 (2) If there is no transition in between $\hat{S}_s$ and $\hat{S}_e$ that
     precedes $T$ in a path of the lattice, insert $T$ between
     them.
 (3) Otherwise,
     (a) Create a new state $S_n$.
     (b) Change the destination state of all transitions
         originating from state $\hat{S}_s$ to $S_n$.
     (c) Insert $T$ between $S_n$ and $\hat{S}_e$.
4. Insert epsilon transitions in between every two
   consecutive states.
```

Fig. 2. The pivot algorithm.

Each transition overlapping *ts*(*T*) is either an *ally* of *T* if it has the same word label *w* as *T* or a *competitor* of *T* otherwise. We sum all the posterior probabilities of the allies of transition *T* and we obtain what we call the posterior probability of word *w*. The rationale of this operation is to avoid missing the important contribution of the same word, often due to small differences in the time alignment, as will be shown in Section 2.2.1. We group the competitors in a similar way and insert into the WCN, as alternatives of *T*.

There are four important components of the pivot algorithm:

(1) The time information computed using the frame numbers is not necessary but is beneficial for the overall performance. In the absence of time information, we use approximate state locations, computation of which is described later in this section.
(2) The algorithm allows the definition of a predefined chunk structure for the final lattice.
(3) The algorithm operates on both weighted and unweighted lattices. In the case of unweighted lattices, various strategies (like the ratio of paths which go through each transition) may be applied to assign scores to WCN transitions.
(4) The labels on the graph are generic and could be words as well as part of speech tags or parse tags.

The algorithm is summarized in Fig. 2 and an extended description of its steps is given below. The algorithm gets as input a word graph (lattice) and generates a WCN:

1. If the lattice is weighted, we first compute the posterior probabilities of all transitions in the word graph, by doing a forward and a backward pass through the graph. The posterior probability of each transition is the sum of the normalized probabilities of all paths[2] in the lattice that go through that transition (Mangu et al., 2000). The normalization is needed to ensure that the sum of the probabilities of all paths in the lattice is equal to 1. In the case of unweighted lattices, we skip this step, but it is also possible to consider each path as equally probable, and proceed with the algorithm.
2. We then sample a sequence of states that lie on a path, from the lattice, to use as the pivot when forming the WCN. The pivot can be the best path or the longest path of the lattice, as well as any random path. In most of our experiments, we either used the best or the longest path.The states on the pivot are assumed to inherit their state time information from the lattice. In our algorithm, the time information is not necessary, but beneficial for the overall performance.

[2] A path is a sequence of adjacent state transitions in the lattice, from the initial state to a final state.

3. To compute the sum of the posterior probabilities of all transitions labeled with word w, that correspond to the same instance, we traverse the lattice in topological order,[3] and insert all transitions into the pivot, finally forming the WCN. For each transition T, we find the two consecutive states $\widehat{S}_{\mathrm{s}}$ and $\widehat{S}_{\mathrm{e}}$ on the pivot that have the maximum overlap, $\mathrm{O}(T, \widehat{S}_{\mathrm{s}}, \widehat{S}_{\mathrm{e}})$, with T:

$$(\widehat{S}_{\mathrm{s}}, \widehat{S}_{\mathrm{e}}) = \operatorname*{argmax}_{S_{\mathrm{s}}, S_{\mathrm{e}}} \mathrm{O}(T, S_{\mathrm{s}}, S_{\mathrm{e}}), \tag{2}$$

$$= \operatorname*{argmax}_{S_{\mathrm{s}}, S_{\mathrm{e}}} (\min(t(S_{\mathrm{e}}), t(S^T_{\mathrm{e}}))) - (\max(t(S_{\mathrm{s}}), t(S^T_{\mathrm{s}}))). \tag{3}$$

We check if there is a transition already inserted between $\widehat{S}_{\mathrm{s}}$ and $\widehat{S}_{\mathrm{e}}$ that precedes T on a path in the lattice. If there is no such transition, we check if another transition with the same label already occurs in between those two states. In the presence of such a transition, we increment its posterior probability by the posterior probability of T. In the absence of a transition with the same label, we create a new transition from $\widehat{S}_{\mathrm{s}}$ to $\widehat{S}_{\mathrm{e}}$, with the label and the posterior probability of T (step 3.2).Otherwise, if there is a transition already inserted between $\widehat{S}_{\mathrm{s}}$ and $\widehat{S}_{\mathrm{e}}$ that precedes T on a path in the lattice, we insert a new state between $\widehat{S}_{\mathrm{s}}$ and $\widehat{S}_{\mathrm{e}}$ (step 3.3.a). We assign a time to the new state S_{n}, $t(S_{\mathrm{n}}) \in (t(S_{\mathrm{s}}), t(S_{\mathrm{e}}))$. We adjust all the transitions, by making them point to the newly inserted state (step 3.3.b), and insert T in between the newly created state and the destination state (step 3.3.c). In the current implementation, we have assigned the newly inserted state, the mean of the times of source and destination states as the state time.
4. Similar to the previous work (Mangu et al., 2000), in order to correct spurious insertions due to misalignments, we insert epsilon transitions in between every two consecutive states. The posterior probability of the epsilon transition is computed so that the sum of the posterior probabilities of all transitions in between these states is equal to 1.

When the state time information is not available in the lattice, we assign each state of the lattice its approximate location on the overall lattice. According to this, the initial state is assigned a time 0, the final states that do not have any outgoing transition are assigned a time 1. All the other states in between are assigned a real number in (0,1), obtained by dividing the average length of all paths up to that state by the average length of all paths that go through that state. As a result, for any transition T, the start state of T, S^T_{s} has a smaller time value than the ending state of T, S^T_{e} (i.e., $t(S^T_{\mathrm{s}}) < t(S^T_{\mathrm{e}})$). These numbers can be computed by a forward and a backward pass through the lattice. We use these approximate state locations to obtain a time slot $ts(T)$, for each transition T.

The pivot algorithm runs in $\mathrm{O}(n \times k)$ time, where n is the number of state transitions in the lattice, and k is the number of chunks in the resulting structure, which usually is much less than n. For example, if the best path is used as the pivot baseline, then k is the length of the best path plus the number of state insertions made. Complexity-wise, it is faster than the sausage algorithm of Mangu et al. (2000) which runs in $\mathrm{O}(n^3)$ time. Recently, the SRILM toolkit (Stolcke, 2002) also includes a new algorithm to build word meshes (confusion networks, sausages) from lattices, that is faster than the original Mangu et al. method (Stolcke, 2004).

2.2. Evaluation of the pivot algorithm

We performed a series of experiments to test the quality of the WCNs and the confidence scores on them. For these experiments, we used a test set of 2174 utterances (31,018 words) collected using the AT&T Spoken Dialog System used for customer care. In this application, the users of the system are greeted by the open ended prompt of *How May I Help You?* (Gorin et al., 1997). The system then tries to understand the responses and act upon them. The language models used in all our experiments are trigram models based on Variable Ngram Stochastic Automata (Riccardi et al., 1996). The acoustic models are sub-word unit based, with triphone context modeling and variable number of Gaussians (4–24). The word accuracy of our test set when recognized with these models (with a relatively smaller beam width) is 66.2%, and the oracle accuracy of

[3] In a topologically ordered lattice, the states of the lattice are ordered such that every transition from a state numbered i to a state numbered j satisfies $i < j$.

the output lattices is 85.7%. This can be considered as an upper-bound on the word accuracy that can be obtained using these lattices.

2.2.1. *Word confidence scores*

To assess the quality of word confidence scores on WCNs, we consider a binary classification task in which we try to reject the misrecognized words, using a rejection threshold for word confidence scores. We plot curves of false rejection vs. false acceptance for this task using various confidence scores as threshold. False rejection is the percentage of words that are correctly recognized in the ASR output, but are rejected as their confidence score is below the threshold. False acceptance is the percentage of words that are misrecognized but are accepted as their confidence score is above that same threshold. When comparing two different sets of confidence scores, usually equal error rates (EERs) are used, which are the points on each curve, where false acceptance rate is equal to the false rejection rate.

In Fig. 3, we have plotted the curves for four different types of confidence scores: the posterior probabilities of the transitions on the best path of the lattice computed during the first step of the pivot algorithm (ASR 1-best), the most likely path of the WCNs using approximate time information, consensus hypotheses of sausages (WCN output of Mangu et al. (2000)'s algorithm), and the most likely path of the WCNs using state time information. Both WCNs and sausages result in better confidence scores than the naive approach of using the posterior probabilities on the best path of the lattice. Although the WCNs using state time information were generated in much less time than sausages, their curve is almost overlapping with the one obtained using sausages. When the time information is not available, the curve for the WCNs is only slightly worse than the one obtained using state times.

Another method for testing the quality of the confidence scores is checking the percentage of correctly recognized words for given confidence scores. One may expect k% of the words having the confidence score of $k/100$ to be correct. Fig. 4 shows our results using confidence score bins without using time information. As seen, the percentage of correctly recognized words in each confidence score bin increases almost monotonically as the confidence score increases.

2.2.2. *Word confusion networks*

The oracle accuracy of the WCNs in our test set is 86.7%, which is slightly more than the oracle accuracy of the lattices. This is because the alignment process creates new paths, increasing the chance of having the correct reference hypothesis in the WCNs. To assess the quality of the WCNs, we have computed oracle accura-

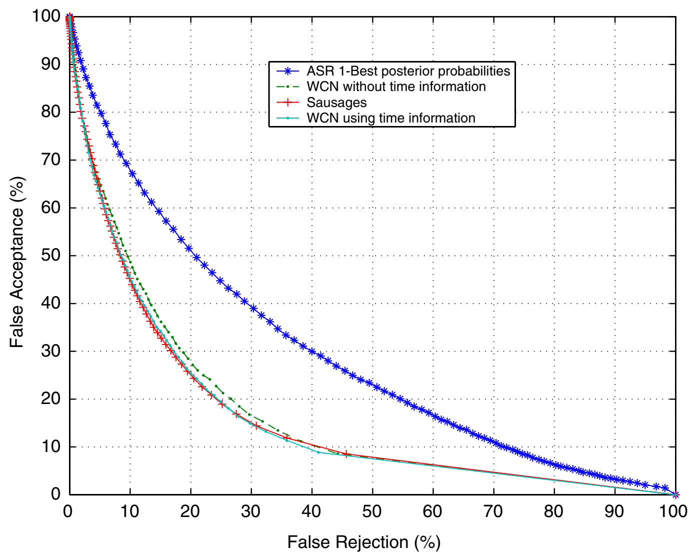

Fig. 3. False rejection vs. false acceptance curves for various confidence scores.

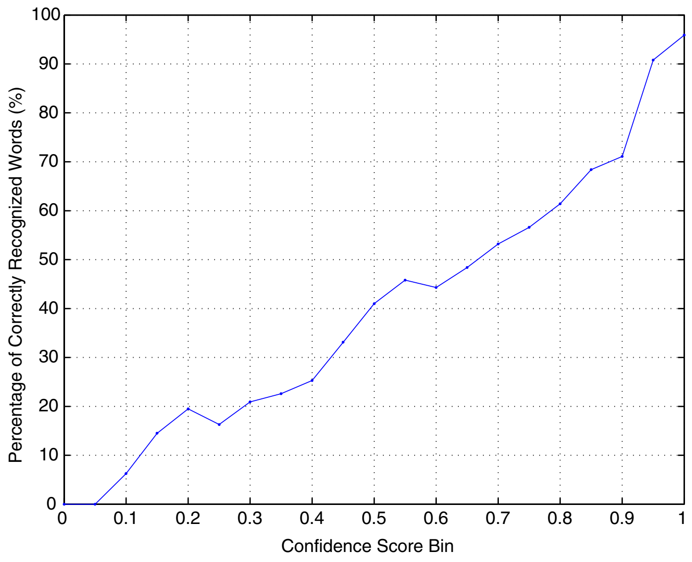

Fig. 4. Percentage of correctly recognized words in confidence score bins on the best path of WCNs computed without using state time information.

cies after pruning the WCNs with two different criteria. The pruning eliminates very low probability transitions, which are very unlikely to contribute to spoken language understanding.

In Table 1, the oracle accuracies after pruning the WCNs by using a threshold for posterior probability are presented. Any arc which has a posterior probability less than k has been pruned from the WCN, then the oracle accuracy has been computed on the pruned WCN.

In Table 2, the oracle accuracies after pruning the WCNs using the rank of the transitions are presented. In between any two states connected by a transition, only the top l transitions that have the highest posterior probability has been retained when computing the oracle accuracy. For example, if we use only the two transitions that have the highest posterior probabilities in between every two states, we can achieve an oracle accuracy of 75.5%. These numbers indicate that, even though we prune the word confusion networks significantly, we still get a high oracle accuracy, showing that the correct hypotheses words get a higher confidence score, and can contribute to spoken language understanding. Using the top candidates in the WCNs, instead of just the ASR 1-best hypothesis, it is possible to be more robust to ASR errors.

Table 1
Oracle accuracies when we pruned all transitions having a posterior probability less than k

k	Oracle accuracy (%)
0.4	67.5
0.2	70.8
0.1	74.2
0.05	77.0
0.01	81.2
0	86.7

Table 2
Oracle accuracies when we took only the most probable l candidates in between all states

l	Oracle accuracy (%)
1	66.2
2	75.5
3	79.0
4	80.9
∞	86.7

Table 3
Effect of pruning on the quality of word confidence scores

ASR time (×real time)	Word accuracy (%)	WCN computation time (×real time)	EER (%)
0.075	35.7	0.0027	45.0
0.138	52.9	0.0045	38.0
0.275	62.9	0.0050	27.2
0.365	65.4	0.0078	25.0
0.482	68.2	0.0130	23.4
0.813	70.1	0.0386	22.5

The run-times are given as fractions of the real time duration of user utterances.

The sizes (number of states and transitions) of the WCNs are much smaller than the sizes of corresponding lattices. In our tests, the size of the WCNs is 7% of the size of the lattices.

2.2.3. *ASR run-time vs. EER*

During ASR, the search space is constrained in order to speed-up the process, compromising word accuracy. We ran experiments to see how much pruning of the search space and output lattices affect the reliability of the confidence scores on the best path of the WCNs. Table 3 shows the ASR run-times, relative to the real-time duration of the utterances in the test set, the corresponding word accuracy of ASR 1-best hypotheses, the computation time for WCNs from the resulting lattices, and equal error rate of the word confidence scores extracted from those WCNs. The word accuracy as well as the quality of the confidence scores improve as we increase the size of search space, and both converge to an upper value. The WCN computation time also increases as we increase the size of the lattices, but even at the lowest EER, the run-time is less than 4% of the real-time utterance length.

3. Named entity extraction

Following the Message Understanding Conference (MUC) terminology (MUC-7, 1998), we call the information items, independent from the call-types, which are contained in the user's request, as *named entities*. They generally refer to proper-name identifiers, time identifiers, and numerical expressions.

In this study, we focus on two specific NEs, which can be found in a large number of human–machine spoken dialog applications: phone numbers and dates.

3.1. *Representing named entities*

NE detection methods can follow either a rule-based approach or a statistical-based approach. Rule-based approaches usually represent NEs by means of regular grammars, which are well suited for encoding numerical NEs or dates as they usually follow a set of fixed patterns. Another advantage of this technique is the possibility of representing the grammar by a Finite State Transducer that can output a value for each expression parsed and accepted. For example, the same normalized value `07/02/2000` corresponds to the following two expressions: `the second of July two thousand` and `July the second two thousand`.

Even though this method gives very good results on written text, the performance decreases when the text to process is the output of a speech recognizer. This is due to a lack of robustness towards speech recognition errors and spontaneous speech effects like edits, pauses, and repairs.

Statistical NE taggers are mostly used for processing ambiguous NEs such as organization names which are not easily modeled by hand written rules. They can also be used for any kind of NEs with very good results if enough training data is available. Because this kind of methods are able to accept and score any kind of word strings, they are much more robust to ASR errors and spontaneous speech effects than regular grammars. Furthermore, the noise generated by these errors can be explicitly modeled by training the tagger on a corpus containing such phenomena. However, since in a Spoken Dialog context a normalized value has to be extracted from each NE detected, an extraction process has to be added to the tagging process.

3.2. Method proposed

The approach proposed follows the method described in Béchet et al., 2002 that integrates both statistical taggers and regular grammars in a 2-step approach. The aim of this method is to propose a new architecture for dialog systems that moves the automatic transcription process from the ASR to the SLU module. The output of the ASR is now a word lattice and it is the understanding module that is in charge of determining the word strings used.

For example, the following sentence:

```
I was charged for calling 1 2 3 4 5 6 7 8 9 0 and ...
```

can be misrecognized by the speech recognizer as:

```
I was charged for calling 1 2 3 for the 6 7 8 9 0 and ...
```

In this case, the 10-digit phone number is split into 2 sequences of digits. This error prevents the correct extraction of the phone number value if this word string is the only input of the SLU module.

In our model, a first understanding process guesses what kind of named entities are likely to occur in the utterance to process. Then some local models specific to each kind of NEs are applied on the speech areas previously selected. These two processes are briefly described in the following sections.

3.2.1. Detection and understanding module

In our system, the output of the ASR module is a word graph. In Section 3.3, we will discuss the advantages of turning this graph into a word confusion network as described in Section 2. In order to detect NE occurrences in the graph, we first apply a statistical NE tagger on the best path of the graph. This tagger is trained on a hand-labeled corpus that contains tags for specifying the beginning and the ending of all NE expressions.

For example, the sentence given as example in the previous section is labeled as: `I was charged for calling <phone> 1 2 3 4 5 6 7 8 9 0 </phone> and`.

This corpus is then aligned with the automatic transcription obtained from our speech recognizer in order to model the possible distortions which can occur during the recognition process. On the previous example we obtain: `I was charged for calling <phone> 1 2 3 for the 6 7 8 9 0 </phone> and`. Finally, the tagger is trained on such a corpus. See Béchet et al. (2002) for more details on the NE tagger.

Once the tagger is applied to the best path of a word graph, each NE detected is associated with its beginning and ending state in the graph corresponding to the first and last word of the NE expression. The extraction process uses these states to limit the search to the sub-graph located between them.

3.2.2. Transcription and extraction module

Once the understanding module has made some hypotheses about the content of a given sentence, thanks to the NE tagger, some local models are applied to the sub-graphs corresponding to each NE detected in order to output a transcription. These local models are linear context-free grammars coded in a finite state machine (FSM) format. These grammars can be either hand-written or data induced from a hand-labeled corpus. In the experiments reported in this work we used a hand-written grammar containing 44 rules for representing standard dates and phone numbers and a set of 400 rules automatically induced from our training corpus.

These grammars are not stochastic. They all represent a valid utterance of a given NE expression. The scoring of the different matches that can be obtained by applying a grammar to the word graph is done by mean of the confidence scores attached to each word of the graph.

The transcription process of a NE detected by the tagger consists of composing the sub-graph extracted by the tagger with the FSM coding the grammar related to the NE detected. By extracting the N-best paths in the composed FSM, we obtain several transcriptions corresponding to several possible values for the NE in the word graph.

The extraction process is a by-product of the transcription process. Indeed, the FSMs coding the grammars contain transduction information in order to output a formatted value for each expression accepted. For example, the date expressed by the two sentences:

```
the tenth of June two thousand and five
June the tenth two thousand and five
```

are recognized by the following grammar:

```
$date → the/<eps> $ordinal/DD of/<eps> $month/MM
        $digit/Y thousand/YY and/<eps> $digit/Y
$date → $month/MM the/<eps> $ordinal/DD
        $digit/Y thousand/YY and/<eps> $digit/Y
```

where `<eps>` is an epsilon symbol, `$ordinal` any ordinal number, `$month` a month name, `$digit` any digit symbol. On the output symbols, `D` is a digit corresponding to a day, `M` a digit of a month and `Y` a digit of a year. The first sentence is accepted by the first grammar rule and produces the following transduction (without the epsilon symbols): `tenth/DD June/MM two/Y thousand/YY five/Y`. It means that the two digits corresponding to the day have to be extracted from the ordinal number `tenth`, the two digits of the month from the month name `June`, the first digit of the year from the digit `two`, the following two digits from `thousand` and the last one from `five`. It leads to the expression: `10/DD 06/MM 2/Y 00/YY 5/Y`. Similarly, the second sentence is accepted by the second grammar rule and produces the transduction: `06/MM 10/DD 2/Y 00/YY 5/Y`. The last step in the value extraction consists in building the formatted values from the transductions obtained. This is straightforward thanks to the tags attached to each digit. For example, because the format `YEAR/MONTH/DAY` was chosen for representing the dates, the two sentences given as example both lead to the same value: `2005/06/10`.

3.3. 1-Best strings, word lattices and word confusion networks

Even if the method previously presented is designed to process word lattices, it can also be applied to single string of words by turning them into FSMs containing only one possible path. This allows us to compare the results obtained by the NE processing module with various output from the speech recognizer. Our point is to show that the understanding process of an utterance should be done *before* the automatic transcription process that outputs the 1-best word string.

Indeed, the 1-best string is calculated by means of probabilistic models which maximize globally the probability of finding the correct sentence. By having to perform equally well on every part of a speech input, the ASR models are not always able to characterize some local phenomenon properly. For example, it has been shown in Rahim et al. (2001) that having a specific model for recognizing *numeric language* in conversational speech improves the performances significantly. However, such a model can be only used to decode answers to a direct question asking for a numerical value, and, in a user-initiative dialog context, we do not know in advance what kind of language the caller is going to use.

By keeping a certain search space in the understanding module, we want to improve the understanding performance measured through the NE detection and extraction tasks. A similar approach is presented in Goel and Byrne (2000). A Minimum Bayes-risk method is used to re-score an *N*-best list of string hypothesis in order to maximize the F-measure explicitly traditionally used for NE evaluation. In this case, the search space is limited to an *N*-best list and a slight improvement in the F-measure is obtained.

In our case, we decided to increase the search space to the whole word lattice or a compact version of it like the WCN structure presented in Section 2.

3.4. Experiments in NE detection and extraction

The experiments reported in this section are made on the AT&T Spoken Dialog corpus for two kinds of NE tags: dates and phone numbers. We chose these NEs in this study because they occur a large number of times in the corpus and they are not dependent on the context of the application. The phone numbers can occur in two kinds of sentences: answers to a direct prompt like `Please give me your phone number starting with the area code`, or embedded in a sentence without being asked. In the tables presenting the evaluation results, the first case is called as *phone(1)* and the second case is called as *phone(2)*. The complexity of the recognition task differs according to the case considered: for *phone(1)* the sentence is very likely to contain only digits and the average sentence length is 10.9 words; for *phone(2)* the phone number is embedded in a sentence and the average sentence length is 39.2 words.

For the case *phone(1)*, the test corpus is made of 617 sentences including 511 phone numbers. For *phone(2)* and the tag *date*, we use a 26K utterance test corpus representing 8.8K dialogs. These sentences contain 607 phone numbers and 465 dates.

We consider two tasks: NE detection and NE extraction. The detection task consists of detecting the occurrence of a NE expression from a given type regardless of its value, in an utterance. This information is very useful in a dialog application, for the Call Classification module (see Section 4) and the Dialog Manager. The extraction task associates to the detection process the extraction of the correct values of the NE expressions detected. The values have to be formatted, as presented in Section 3.2.2, and an extracted value is considered correct if its formatted value is identical to the reference one. We call this measure the *understanding accuracy*.

The results are given according to the following standard measures: precision *P*, recall *R* and F-measure *F*. They are defined as follows:

$$P = \frac{\#\text{ of correct answers}}{\#\text{ of answers}}, \tag{4}$$

$$R = \frac{\#\text{ of correct answers}}{\#\text{ total of tags in the reference corpus}}, \tag{5}$$

$$F = \frac{2 * R * P}{R + P}. \tag{6}$$

3.4.1. Detection results

The detection results in Table 4 clearly show that a very significant gain can be achieved by using word graphs instead of single strings as input of the understanding module: an average absolute gain of 8.6% in F-measure is obtained between *1-best* strings and WCNs for the three kinds of NEs.

Table 4
NE detection results for phone numbers and dates using ASR 1-best path (1-best), word lattices, and WCNs

	1-Best	Lattice	WCN
Phone(1)			
Precision	99.5	98.7	99.3
Recall	75.7	89.2	88.5
F-measure	86	93.7	93.6
Phone(2)			
Precision	98	94.2	94.8
Recall	67.2	81.3	81.4
F-measure	79.8	87.3	87.6
Date			
Precision	94.8	76.7	90.3
Recall	47.3	66.23	62
F-measure	63.1	71	73.6

Both word graph inputs (lattices and WCNs) give roughly the same results but it is important to point out that the detection process on the whole word lattices is much slower than the same process on the WCNs. This is due to the difference of size between these different structures.

Indeed, NE extraction from lattices took 0.42 times the real-time duration of the audio files (which does not leave much time for further processing in a real system) and from WCNs took 0.017 times, about 25 times faster. In these experiments, we used the lattices and WCNs on the fourth row of Table 3, therefore, the WCN computation overhead is only 0.0078 times real-time. Even with the WCN computation time, NE extraction from WCNs is much faster than lattices, without degrading performance.

3.4.2. Extraction results

The extraction results presented in Table 5 are contrasted: for the date entities, a very significant gain in F-measure is obtained by using word graphs, and especially WCNs, instead of 1-best strings; however, despite the gain in F-measure for the detection of phone number entities, there is no gain in the extraction F-measure for *phone(1)* and *phone(2)* entities with lattices and even some loss with WCNs.

A closer look to these results shows that if the F-measure is not improved for phone number extraction, a significant gain in the recall measure both for lattices and WCNs (6% absolute gain with lattices and 4% absolute gain with WCNs for phone(1)) is achieved. This means that more correct telephone values are extracted with the word graphs. However, at the same time, a lot of the phone numbers correctly detected according to the results obtained on NE detection are associated with incorrect values.

This can be explained by the following remark: the phone numbers detected in the word graphs and not in the 1-best strings are very likely to contain errors as the most probable words according to the general models are not the correct ones. This indicates a noisy recognition and the probability of having at least one error in the 10 digits of each of them is quite high. Therefore it is interesting to consider not only the first value given by the NE extraction process but the *N* first values (*N*-best). Let's point out that this situation is realistic in a spoken dialog context as it is often possible to use some contextual information in order to disambiguate several outputs (for example, a directory containing valid phone numbers).

This is illustrated by Fig. 5 that shows the F-measure results for different size of *N*-best lists, from 1 to 10, for the tag *phone(1)*. Four different inputs are compared:

(1) `1-best`: best strings of words output by the ASR module;
(2) `lattice (x25)`: word lattices with no pruning (25 times slower than WCNs);
(3) `lattice (x6)`: word lattices pruned in order to be only 6 times slower than the corresponding WCNs;
(4) `WCN`: word confusion network.

As we can see, word graph inputs outperform significantly 1-best string inputs if more than one value is kept for each entity detected. For example, by keeping the first 3 values, we obtain an absolute gain of 5% in the F-measure with WCNs instead of 1-best strings.

Table 5
NE extraction results for phone numbers and dates using ASR 1-best path (1-best), word lattices, and WCNs

	1-Best	Lattice	WCN
Phone(1)			
Precision	91.8	83.9	82.9
Recall	69.9	75.8	73.8
F-measure	79.3	79.7	78
Phone(2)			
Precision	84.9	75	71
Recall	58.2	64.8	61
F-measure	69	69.5	65.6
Date			
Precision	79.7	61.2	78.7
Recall	39.8	52.8	54
F-measure	53	56.7	64.1

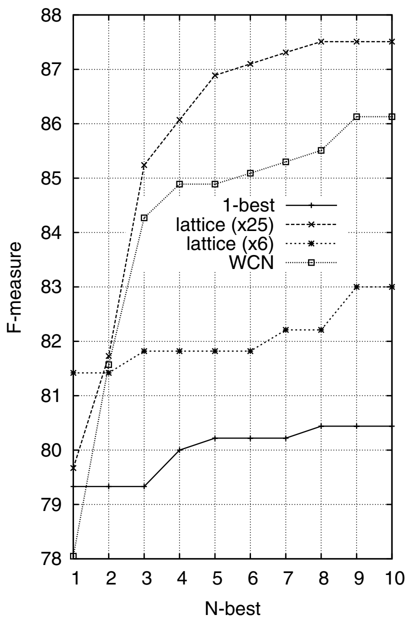

Fig. 5. F-measure vs. the size of the *N*-best list of values for the entity phone number (*phone1*). Comparison between results obtained on 1-best word strings, WCNs and 2 kind of word lattices, one with pruning (x6) and one without pruning (x25).

The comparison between WCNs and lattices highlights the benefits obtained with the method presented in this paper over a simple pruning of the word lattices: even though full lattices (x25) give better results than WCNs, the time needed to process them makes them difficult to use in a real application. Pruning lattices is not a solution as Fig. 5 shows that the performance decreases very significantly. Therefore, WCNs are a good way of compacting lattices for NE detection and extraction without decreasing the performance.

4. Call classification

In goal driven human–machine dialog, the other key component of the understanding process is determining the intent of the speaker. This step can be framed as a multi-label multi-class classification problem for goal-oriented call routing systems (Haffner et al., 2003; Natarajan et al., 2002; Kuo et al., 2002; Schapire et al., 2002). We assume that the call-types are pre-defined and applicable to all utterances in the conversation.

In this work, we exploit the error estimates from ASR and multiple hypotheses encoded in a WCN to improve the user intent classification, making it more robust to ASR errors.

In our previous work, we have presented methods to exploit word confidence scores (Rose et al., 2001; Tur et al., 2002) and WCNs (Tur et al., 2002) for a Bayesian classifier with feature pre-selection.

A simple Bayesian classifier tries to estimate the posterior probabilities using the well-known Bayes formula

$$P(C_i|W) = \frac{P(W|C_i) \times P(C_i)}{P(W)},$$

where C_i is a call-type and W is the utterance (or ASR 1-best). Instead of using all words in the utterance, selecting some salient phrases, $f_i(W)$, among possible features (n-grams) beforehand for each call-type, C_i, is reported to greatly improve the performance (Gorin et al., 1997). The posterior probability, $P(C_i|f_i(W))$, is thus computed just using those features. It is then straightforward to embed the ASR word confidence scores during run-time in this framework. We have simply estimated the salient feature weights from the word confidence scores and adjusted the posterior probability accordingly

$$P(C_i|f_i(W)) \times P(f_i(W)).$$

In that previous work, we have reduced the call classification error rate for a given false rejection rate about 20% relatively. Obviously, if the salient feature selection step is skipped and all words in the utterance are used for all call-types, then this method will not work since the same words are considered for all call-types, and the probability of each call-type is scaled by the same factor.

In this study, we employ an off-the-shelf discriminative classifier, namely Boosting, without assuming any feature pre-selection. In this section, we propose algorithms for exploiting WCNs for call classification. We suggest methods independent of the classifier used and a method with a new formulation for Boosting family of classifiers. The mathematical formulation of the algorithm allows to exploit confidence scores from a ASR 1-best output or from WCNs.

4.1. Exploiting word confusion networks

We propose various methods to exploit WCNs. In all of them the main idea is to extend the feature space of the classification algorithm using the WCNs also considering the word confidence scores. The first method is independent of the classifier used, and serves as a baseline to evaluate other methods. It aims to filter or bin the features used according to the word confidence scores obtained from the WCN. The second and third ones are specific to Boosting. One of them aims to change the weak learner found, and the other one aims to change the way weak learners are used during run-time.

4.1.1. ASR output filtering and binning

One approach for handling ASR errors during semantic classification is filtering out words that have very low confidence scores from the ASR output during run-time. In order to determine this threshold one can use a development test set or select the confidence score for which the percentage of misrecognized words is the same as the correctly recognized words. In this approach one can keep the training as is, and just do the filtering during run-time.

In the cases where the ASR output for the training data is also available, binning the training and test words using their confidence scores is another approach. To implement binning, each word w in the training data is replaced with w_0 if its confidence score is less than or equal to a threshold t_0, and w_i if its confidence score is in $(t_{i-1}, t_i]$, where $t_0 < t_1 < \cdots < t_{i-1} < t_i < \cdots < t_k = 1$.

4.1.2. Extended boosting algorithm

Boosting aims to combine "weak" base classifiers to come up with a "strong" classifier (Schapire and Singer, 1999). This is an iterative algorithm, and in each iteration, a weak classifier is learned so as to minimize the training error. Details of this algorithm can be found in Appendix A. Assume that weak learners make their predictions based on a partitioning of the domain $\mathcal{X}$ into disjoint blocks X_j. An instance would be partitioning with respect to the absence or presence of a feature. Then one can define the output of the Boosting classifier, $f(x, l)$, for each class, l, for a given object, $x \in X_j$, as in Eq. (A.2)

$$f(x,l) = \sum_{t=1}^{T} \alpha_t \times c^t_{jl},$$

where c^t_{jl} is score of the weak learner c_{jl} for iteration t and α_t is the weight of the base learner.

In this work, we employed the Boostexter (Schapire and Singer, 2000) tool and used word n-grams (phrases) as features, where the weak learners partition the domain with respect to the absence or presence of an n-gram. For call classification, where the input of the classifier is the ASR output, it is clear that instead of making the binary decision of absence or presence of a feature, it makes more sense to exploit confidence scores obtained from the word confusion networks. This requires the following extension to the above formula:

$$f(x,l) = \sum_{t=1}^{T} p(j) \times \alpha_t \times c^t_{jl},$$

where $p(j)$ is the probability of being in the partition X_j, estimated by the confidence scores of the WCN. We have used the average of the word confidence scores in that phrase as the feature weights. This is a very similar approach to our previous work on exploiting ASR word confidence scores in a Bayesian classifier with feature selection. In this case, we exploit the fact that Boosting is already selecting features for classification, so we simply weigh them. Note that this change takes place only during run-time, but it is possible to extend the algorithm so that the weak learners consider all the features in WCNs with their confidence scores. Here, we train the weak learners from human transcribed data, which do not contain any confidence scores (i.e., the confidence score of each word is 1).

4.1.3. Boostexter with scored features

Boostexter already has the capability of exploiting the scores associated to the features. These scores can be any real number and is not limited to the interval $[0,1]$. Instead of using two partitions, i.e., absence and presence of a feature, the weak learner tries to come up with three partitions, i.e., absence of a feature, presence of a feature with a score more than some threshold and less than that threshold. This threshold is also learned from training data. Unlike the previous approach, this method changes the model learned when the confidence scores for the training data is available.

4.2. Experiments for call-type classification

In order to evaluate the proposed methods to exploit WCNs for call classification, we have again used the utterances collected from the AT&T Spoken Dialog application. We first describe our data and the evaluation metrics used to compare model performances, then show results.

4.2.1. Data

Table 6 summarizes the amount of data used for training and testing for this task along with the total number of call-types, average utterance length, and call-type perplexity. Perplexity is computed using the prior distribution of the call-types in the training data.

4.2.2. Evaluation metrics

The goal of call classification is to accept (give a confidence higher than some threshold) only the true (according to manual labels) call-types and reject others for a given utterance. In consistent with the machine learning community, while evaluating the classification performance, we used the *recall* and *precision* metrics with micro-averaging considering multiple call-types. Recall, R, is defined as the proportion of all the true call-types that are correctly deduced by the classifier. Precision, P, is defined as the proportion of all the accepted call-types that are also true:

$$R = \frac{\#\text{ of true positives}}{\#\text{ of true call-types}},$$

$$P = \frac{\#\text{ of true positives}}{\#\text{ of accepted call-types}}.$$

4.2.3. Results

The recall vs. precision numbers when we used the transcriptions of utterances during training, and the ASR output, as well as filtered ASR output of the test data is given in Fig. 6. In order to evaluate this approach, we have used the WCN 1-Best string to have word confidences. To select the threshold for filtering words from the ASR output of the test set, we first computed confidence scores for the words in the training

Table 6
Data characteristics used in the experiments

Training data size	9094 utterances
Test data size	5171 utterances
Number of call-types	84
Call-type perplexity	32.64
Average length	10.66 words

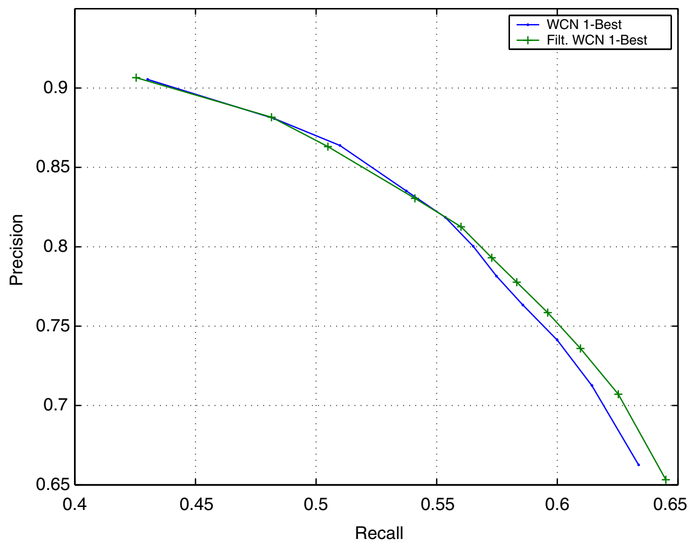

Fig. 6. Recall vs. precision for call classification, when low confidence words are filtered from ASR output.

set. We divided all the words into bins using their scores. We selected the confidence score for which the percentage of misrecognized words is the same as the percentage of correctly recognized words. As a result, 0.63 was selected as the threshold, and 11.4% of the words are filtered from the ASR output of the test data. Filtering low confidence words resulted in slightly better Precision at high Recall levels (see Fig. 6). We also tried other thresholds, but could not obtain better results. Binning the words using their confidence scores did not result in significant improvements, either.

Fig. 7 presents our results using extended Boosting with only WCN 1-best (or consensus hypothesis). Topmost curve is obtained using human transcriptions and bottom-most one using ASR 1-best. In order to show the potential improvement using WCNs, we have first made an experiment by keeping all the *n*-grams of the WCN's best path occurring also in the human transcriptions. This is the upper bound we can get by using WCN 1-best, shown by the curve with circles. The improvement using extended Boosting is shown by the curve just below that. As seen for all thresholds, we obtained better recall and precision values than using ASR 1-best, and we achieved one third of the way to the upper bound.

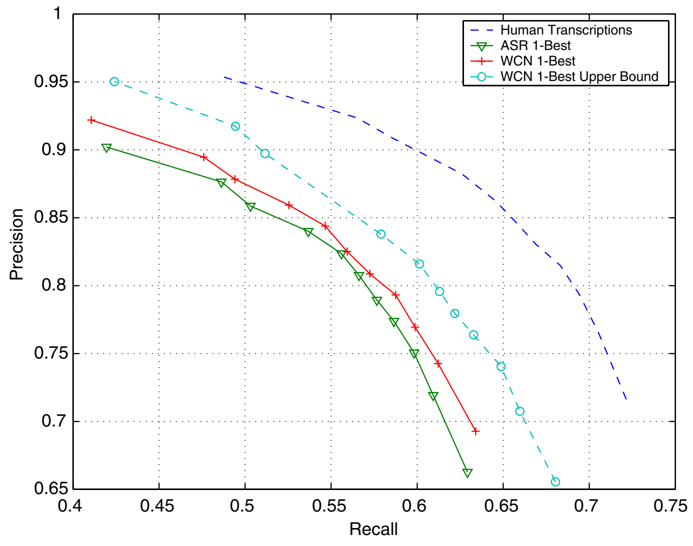

Fig. 7. Results obtained by extending Boosting with feature confidence scores for WCN 1-best.

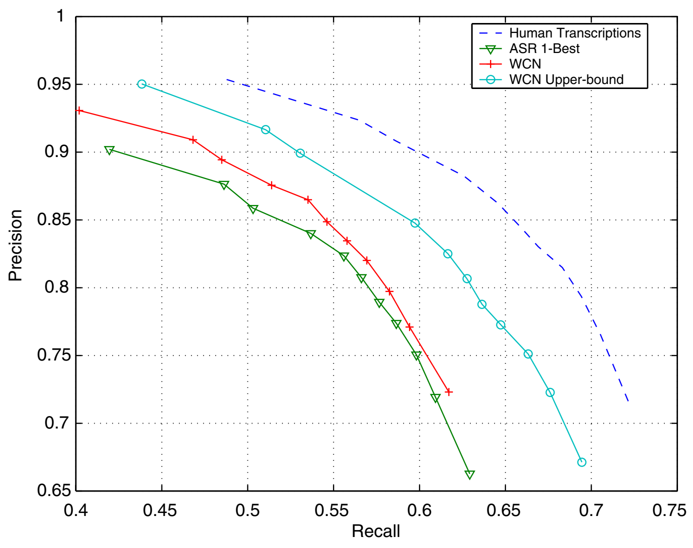

Fig. 8. Results obtained by extending Boosting with feature confidence scores for whole WCN.

Fig. 8 presents our results using extended Boosting using the whole WCNs. Similar to the previous figure, the upper bound is now much closer to the curve obtained using human transcriptions. Exploiting the whole WCNs we have obtained better performance especially on high rejection thresholds. For lower thresholds using just the consensus hypotheses gave better results. This may be due to the *n*-grams found in the WCN increasing the false positives. On the average, for a given recall, we significantly[4] decreased the error rate (1 − precision) by 5–10%.

Using Boostexter with scored features did not help, actually we have got significantly inferior results on the test set although training set error rate was lower. This may be due to lack of training data to generalize or determine the threshold for each weak learner.

5. Conclusions

In this paper, we have shown the benefits of a tighter integration of the speech recognizer with the following spoken language understanding, by using a richer representation to describe an ASR output. We have presented novel algorithms to use WCNs obtained from ASR word lattices instead of ASR 1-best for named entity extraction and call classification tasks. It is also possible to use the proposed methods for other speech processing tasks, such as speech to speech machine translation and information retrieval from speech. We have evaluated these methods using a large corpus of utterances collected using the AT&T Spoken Dialog System for customer care. For named entity extraction, we have improved the accuracy significantly by using both word lattices and WCNs, but processing of WCNs has been 25 times faster than lattices, which is very important for real-life applications. For call classification, we have shown between 5% and 10% relative reduction in error rate using WCNs compared to ASR 1-best output.

Acknowledgments

We thank Robert Schapire and Mazin Rahim for many useful discussions.

[4] Using Z-test at 95% confidence.

Appendix A. Boosting

Boosting aims to combine "weak" base classifiers to come up with a "strong" classifier (Schapire and Singer, 1999). This is an iterative algorithm, and in each iteration, a weak classifier is learned so as to minimize the training error.

The algorithm generalized for multi-class and multi-label classification is given in Fig. A.1. Let $\mathscr{X}$ denote the domain of possible training examples and let $\mathscr{Y}$ be a finite set of classes of size $|\mathscr{Y}| = k$. For $Y \subseteq \mathscr{Y}$, let $Y[l]$ for $l \in \mathscr{Y}$ to be

$$Y[l] = \left\{ \begin{array}{lll} +1 & \text{if} & l \in Y, \\ -1 & \text{if} & l \notin Y. \end{array} \right\}$$

The algorithm begins by initializing a uniform distribution, $D_1(i,l)$, over training examples, i, and labels, l. After each round this distribution is updated so that the example-class combinations which are easier to classify get lower weights and vice versa. The intended effect is to force the weak learning algorithm to concentrate on examples and labels that will be the most beneficial to the overall goal of finding a highly accurate classification rule. The final "strong" learner, H, is then nothing but a linear combination of the individual weak learners, h_t.

Schapire and Singer (1999) have proved a bound on the training error rate, i.e., *Hamming Loss* (HL), of H in this algorithm.

$$\mathrm{HL}(H) \leqslant \prod_{t=1}^{T} Z_t \tag{A.1}$$

and Z_t is the normalization factor computed on round t

$$Z_t = \sum_{l \in \mathscr{Y}} \sum_{i=1}^{m} D_t(i,l) \exp(-\alpha_t Y_i[l] h_t(x_i,l)).$$

HL is defined as the fraction of misclassified examples, i, and labels, l,

$$\mathrm{HL}(H) = \frac{1}{mk} |i,l : H(x_i,l) \neq Y_i[l]| = \frac{1}{mk} \sum_{l} \sum_{i} \delta(x_i,l),$$

where

$$\delta(x_i,l) = \left\{ \begin{array}{ll} 1 & \text{if } H(x_i,l) \neq Y_i[l], \\ 0 & \text{otherwise.} \end{array} \right\}$$

- Given the training data from the instance space X: $(x_1, Y_1), ..., (x_m, Y_m)$ where $x_i \in \mathcal{X}$ and $Y_i \subseteq \mathcal{Y}$, $|\mathcal{Y}| = k$
- Initialize the distribution $D_1(i,l) = \frac{1}{mk}$
- For each iteration $t = 1, ..., T$ do
  - · Train a base learner, h_t, using distribution D_t.
  - · Update

    $$D_{t+1}(i,l) = \frac{D_t(i,l)e^{-\alpha_t Y_i[l] h_t(x_i,l)}}{Z_t}$$

    where Z_t is a normalization factor and α_t is the weight of the base learner.
- Then the output of the final classifier is defined as:

  $$H(x,l) = sign(f(x,l))$$

  where

  $$f(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l)$$

Fig. A.1. The algorithm *Adaboost.MH*.

The bound in Eq. (A.1) is important, because in order to minimize this training error, now it is a reasonable approach to minimize Z_t on each round of boosting. This leads to criteria for finding weak hypotheses, $h(x,l)$, for a given iteration. Assume that weak learners, h, make their predictions based on a partitioning of the domain $\mathscr{X}$ into disjoint blocks X_j. Let $c_{jl} = h(x,l)$ for $x \in X_j$, then Schapire and Singer have proved that the optimal c_{jl} is given by

$$c_{jl} = \frac{1}{2}\ln\left(\frac{W_{+}^{jl}}{W_{-}^{jl}}\right),$$

where $W_b^{jl} = \sum_{i:x_i \in X_j} D(i,l)(1-\delta(x_i,l))$ and the output of the Boosting classifier, $f(x,l)$, for each class, l, for a given utterance, $x \in X_j$, is defined as follows:

$$f(x,l) = \sum_{t=1}^{T} \alpha_t c_{jl}^t, \tag{A.2}$$

where c_{jl}^t corresponds to the weak learner c_{jl} for iteration t.

References

Bangalore, S., Bordel, G., Riccardi, G., 2001. Computing consensus translation from multiple machine translation systems. In: Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). Trento, Italy, pp. 350–354.

Béchet, F., Wright, J., Gorin, A., Hakkani-Tür, D., 2002. Named entity extraction from spontaneous speech in How May I Help You?^{SM}. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). Denver, CO, pp. 597–600.

Cortes, C., Haffner, P., Mohri, M., 2003. Lattice kernels for spoken dialog classification. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Hong Kong, pp. 628–631.

Cox, S., Cawley, G., 2003. The use of confidence scores in vector based call-routing. In: Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH). Geneva, Switzerland, pp. 633–636.

Cox, S., Dasmahapatra, S., 2002. High-level semantic approaches to confidence estimation in speech recognition. IEEE Transactions on Speech and Audio Processing 10 (7), 460–471.

Goel, V., Byrne, W., 2000. Minimum bayes risk automatic speech recognition. Computer Speech and Language 14 (2), 115–135.

Goel, V., Byrne, W., Khudanpur, S., 1998. LVCSRrescoring with modified loss functions: A decision theoretic perspective. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). Seattle, WA, pp. 425–428.

Gorin, A., Abella, A., Alonso, T., Riccardi, G., Wright, J., 2002. How May I Help You? IEEE Computer Magazine 35 (4), 51–56.

Gorin, A., Riccardi, G., Wright, J., 1997. How May I Help You? Speech Communication 23, 113–127.

Gretter, R., Riccardi, G., 2001. On-line learning of language models with word error probability distributions. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). Salt Lake City, UT, pp. 557–560.

Gupta, N., Tur, G., Hakkani-Tür, D., Bangalore, S., Riccardi, G., Rahim, M., To appear. The AT&T spoken language understanding system. IEEE Transactions on Speech and Audio Processing (To appear).

Haffner, P., Tur, G., Wright, J., 2003. Optimizing SVMs for complex call classification. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Hong Kong, pp. 632–635.

Hakkani-Tür, D., Riccardi, G., 2003. A general algorithm for word graph matrix decomposition. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Hong Kong, pp. 596–599.

Hakkani-Tür, D., Riccardi, G., Gorin, A., 2002. Active learning for automatic speech recognition. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). Orlando, FL, pp. 3904–3907.

Hazen, T., Seneff, S., Polifroni, J., 2002. Recognition confidence scoring and its use in speech understanding systems. Computer Speech and Language 16, 49–67.

He, Y., Young, S., 2003. A data-driven spoken language understanding system. In: Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). U.S. Virgin Islands, pp. 583–588.

Johnson, M.T., Harper, M.P., Jamieson, L.H., 1998. Interfacing acoustic models with natural language processing systems. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). Sydney, Australia, pp. 2419–2422.

Kingsbury, B., Mangu, L., Saon, G., Zweig, G., Axelrod, S., Goel, V., Visweswariah, K., Picheny, M., 2003. Toward domain-independent conversational speech recognition. In: Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH). Geneva, Switzerland, pp. 1881–1884.

Kuo, J., Lee, C., Zitouni, I., Fosler-Lusser, E., Ammicht, E., 2002. Discriminative training for call classification and routing. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). Denver, CO, pp. 1145–1148.

Mangu, L., Brill, E., Stolcke, A., 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. Computer Speech and Language 14 (4), 373–400.

MUC-7, April 1998. Proceedings of the 7th Message Understanding Conference (MUC-7). Fairfax, VA.

Natarajan, P., Prasad, R., Suhm, B., McCarthy, D., 2002. Speech enabled natural language call routing: BBN call director. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). Denver, CO, pp. 1161–1164.

Oerder, M., Ney, H., 1993. Word graphs: An efficient interface between continuous-speech recognition and language understanding. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vol. 2. pp. 119–122.

Rahim, M., Riccardi, G., Saul, L., Wright, J., Buntschuh, B., Gorin, A., 2001. Robust numeric recognition in spoken language dialog. Speech Communication 34, 195–212.

Riccardi, G., Hakkani-Tür, D., 2003. Active and unsupervised learning for automatic speech recognition. In: Proceedings of the 8th European Conference on Speech Communication and Technology (EUROSPEECH). Geneva, Switzerland, pp. 1825–1828.

Riccardi, G., Pieraccini, R., Bocchieri, E., 1996. Stochastic automata for language modeling. Computer Speech and Language 10, 265–293.

Rose, R.C., Yao, H., Riccardi, G., Wright, J.H., 2001. Integration of utterance verification with statistical language modeling and spoken language understanding. Speech Communication 34, 321–331.

Schapire, R.E., Rochery, M., Rahim, M., Gupta, N., 2002. Incorporating prior knowledge into boosting. In: Proceedings of the International Conference on Machine Learning (ICML). Sydney, Australia, pp. 538–545.

Schapire, R.E., Singer, Y., 1999. Improved boosting algorithms using confidence-rated predictions. Machine Learning 37 (3), 297–336.

Schapire, R.E., Singer, Y., 2000. Boostexter: A boosting-based system for text categorization. Machine Learning 39 (2/3), 135–168.

Stolcke, A., 2002. SRILM - an extensible language modeling toolkit. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). Denver, Colorado, pp. 901–904.

Stolcke, A., 2004. SRILM - version history. http://www.speech.sri.com/projects/srilm/docs/CHANGES.

Stolcke, A., Konig, Y., Weintraub, M., 1997. Explicit word error rate minimization in *N*-best list rescoring. In: Proceedings of the 5th European Conference on Speech Communication and Technology (EUROSPEECH). Rhodes, Greece, pp. 163–166.

Tur, G., Wright, J., Gorin, A., Riccardi, G., Hakkani-Tür, D., 2002. Improving spoken language understanding using word confusion networks. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). Denver, CO, pp. 1137–1140.

Wang, Y., Acero, A., Chelba, C., 2003. Is word error rate a good indicator for spoken language understanding accuracy. In: Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). U.S. Virgin Islands, pp. 577–582.

Wessel, F., Macherey, K., Ney, H., 1999. A comparison of word graph and n-best list based confidence measures. In: Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH). Budapest, Hungary, pp. 315–318.