Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Speech Communication 42 (2004) 207–225
SPEECH COMMUNICATION
www.elsevier.com/locate/specom

Detecting and extracting named entities from spontaneous speech in a mixed-initiative spoken dialogue context: How May I Help You?[sm,tm]

Frédéric Béchet [a,*], Allen L. Gorin [b], Jeremy H. Wright [b], Dilek Hakkani Tür [b]

[a] *LIA, University of Avignon, BP1228, 84911 Avignon Cedex 09, France*
[b] *AT&T Labs, 180 Park Avenue, Florham Park, NJ 07932, USA*

Received 17 October 2002; received in revised form 14 June 2003; accepted 1 July 2003

Abstract

The understanding module of a spoken dialogue system must extract, from the speech recognizer output, the kind of request expressed by the caller (the *call type*) and its parameters (numerical expressions, time expressions or proper-names). Such expressions are called Named Entities and their definitions can be either generic or linked to the dialogue application domain. Detecting and extracting such Named Entities within a mixed-initiative dialogue context like *How May I Help You?*[sm,tm] (HMIHY) is the subject of this study. After reviewing standard methods based on hand-written grammars and statistical tagging, we propose a new approach, combining the advantages of both in a 2-step process. We also propose a novel architecture which exploits understanding to improve recognition accuracy: the output of the Automatic Speech Recognition module is now a word lattice and the understanding module is responsible for transcribing the word strings which are useful to the Dialogue Manager. All the methods proposed are trained and evaluated on a corpus comprising utterances from live customer traffic.

Résumé

Les systèmes automatiques de dialogue téléphonique contiennent généralement un module de compréhension chargé de traiter les sorties du module de reconnaissance automatique de parole. Ce traitement consiste à extraire non-seulement le type de requête exprimée par l'utilisateur mais aussi les paramètres de cette requête tels que les expressions numériques, temporelles ou bien encore les noms propres. Ces expressions sont généralement appeleés des Entités Nommées et leurs définitions peuvent être génériques ou bien liées à un domaine d'application particulier. Détecter et extraire de telles entités dans le cadre d'un système automatique de dialogue téléphonique à initiative mixte tel que *How May I Help You?*[sm,tm] (HMIHY) est le sujet de cette étude. Après avoir passé en revue les méthodes habituelles basées sur des grammaires écrites manuellement ou bien sur des étiqueteurs statistiques, nous proposons une nouvelle approche permettant de combiner leurs avantages respectifs. Nous proposons également une nouvelle architecture, pour les systèmes automatiques de dialogue téléphonique, qui utilize les résultats du module de compréhension afin

* Corresponding author. Tel.: +33-4-90-84-35-12; fax: +33-4-90-84-35-01.
E-mail addresses: frederic.bechet@lia.univ-avignon.fr (F. Béchet), algor@research.att.com (A.L. Gorin), jwright@research.att.com (J.H. Wright), dtur@research.att.com (D. Hakkani Tür).

0167-6393/$ - see front matter
doi:10.1016/j.specom.2003.07.003

d'améliorer la transcription des requêtes des utilisateurs. Toutes les méthodes proposées sont évaluées sur un corpus contenant de réels dialogues entre des utilisateurs et une application mise en service sur une large échelle.

Keywords: Speech recognition; Spoken dialogue systems; Spoken language understanding; Named entities

1. Introduction

Interactive spoken dialogue systems are now employed in a wide range of applications, such as directory assistance or customer care, on a very large scale. Dealing with a large population of non-expert users has two major consequences: on one hand, there is great variability in the spontaneous speech being processed, requiring a very high robustness from every part of a dialogue system; on the other hand, the large amount of *real* data collected through these spoken dialogue systems raises new issues and makes possible the use of even more automatic learning and corpus-based methods at each step of the dialogue process. The kind of spoken dialogue system considered in this paper can be viewed as an interface between a user and a database. One of the roles of the Dialogue Manager (DM) module is to determine, firstly what kind of query the database is going to be asked and secondly with which parameters. In the *How May I Help You*^{sm,tm} (HMIHY) customer care corpus we used in this study (Gorin et al., 1997), if a user wants his account balance, the query will be *accessing the account balance field of the database* with the customer identification number as the parameter.

Such database query types are denoted *call-type* and their parameters are the information items, independent from the call type, which are contained in the user's request. They are often called *Named Entities* (NEs) as in the evaluation programs *Message Understanding Conference* (MUC), *DARPA HUB-4* or *Automatic Content Extraction* (ACE). The most general definition of a named entity is the following: a sequence of words that refers to a unique identifier. More precisely, NEs can refer to:

- proper name identifiers, like organization, person or location names;
- time identifier, like dates, time expressions or durations;
- quantities and numerical expressions, like monetary values, percentage or phone numbers.

The definition of such tags for the MUC evaluation program can be found in (Chinchor and Robinson, 1998). In the framework of spoken dialogue systems, this classification is more complex and moreover dependent on the application targeted. Indeed, *generic* NEs like *dates* or *amounts* need to be enriched with semantic information related to their functions within the dialogue. For example, if a *date* is detected, the Dialogue Manager needs to know which object is attached to this date (e.g. date of a phone call or date of a bill). These NEs will be referred to as *dialogue-dependent* NEs in this paper. Detecting and extracting such dialogue-dependent NEs as well as generic ones is the subject of this study.

Our goal is to bridge the gap between the transcription and the understanding processes in spoken dialogue system architectures. Traditionally the Automatic Speech Recognition (ASR) module outputs a single string of words (called the *1-best* string) which is processed by the Natural Language Understanding (NLU) module. In the model proposed, the output of the ASR module is a word graph on which the NE detection and extraction are performed. Therefore, understanding and transcribing an utterance is now a single process.

The method proposed is also an attempt to merge a data-driven and a knowledge-based technique, both of which have been widely used for NE processing: a rule-based approach and a statistical tagger approach. By taking advantage of the high precision provided by rule-based techniques and the high recall that can be achieve with statistically-based techniques, we obtain significant improvement in the F-measure score for NE de-

tection compared with each method taken separately. For NE extraction, we are able to improve the understanding accuracy by correcting errors occurring in the 1-best string provided by the ASR module, if we take into account not only the best value extracted for each NE detected, but the first n values (with $n < 5$).

2. Named entities in How May I Help You ?

The corpus we used in this study contains 130k utterances from live customer traffic collected on the *How May I Help You ?* (HMIHY) service. HMIHY is an AT&T customer-care application (Gorin et al., 1997) which deals with requests and complaints from AT&T customers about their phone bills. The dialogue strategy implemented is a mixed-initiative one: the top-level of the dialogue implements a user-initiative dialogue by simply asking the user *How may I help you?*. A short dialogue sometimes ensues for clarifying the request, and finally the user is sent either to an automatic system or to a human representative depending on the availability of such an automatic process for the request recognized.

The only NEs which are manually tagged in the HMIHY corpus are those which can be useful to the Dialogue Manager. Therefore their definitions always contain semantic information related to the dialogue and their context of occurrence.

These tags can be seen as the *interpretations* of the roles of the NEs within the dialogue. For example, the NE tag `Which_Bill` refers to an expression that identifies a customer's bill, like: `my January bill`, `my previous statement` or `bill issued on the second of January`. As can be seen, a date can represent a `Which_Bill` entity, but at the same time a date can be an `Item_Date` which corresponds to the date of a phone call. These kinds of ambiguities, inevitable in a dialogue application, make the NE detection and extraction tasks significantly harder in this context than in a Broadcast News context.

In order to distinguish the difficulties which are due to the intrinsic ambiguities in the NE definitions and those due to the processing of spontaneous conversational speech, we studied four different NE tags in this paper: two *generic* ones, corresponding to the usual definition of NEs and representing phone numbers and dates; two *real* dialog-dependent NE tags, from the HMIHY tag set and representing money expressions referring to a charge written on customers' bills (`Item_Amount`) and the tag `Which_Bill` previously presented. Examples of some occurrences of these tags with their corresponding normalized values are given in Table 1.

Table 1
Examples of NE tags, contexts and values in the training corpus

Tag	Context	Value
`Item_Amount`	This 22 dollar charge	22.00
Phone	386 5715 area code 201	2013865715
Date	June tenth	????/06/10
Which_Bill	Most recent statement	Latest

3. Processing NEs

Two approaches have been proposed for processing NEs: rule-based systems (mainly hand-written rules), like Black et al. (1998) and statistical taggers (implementing a Maximum-Entropy or a Hidden Markov Model approach), like Kubala et al. (1998). Even if these methods have been frequently compared (through the MUC-7 or HUB-4 programs or on spontaneous speech in (Huang et al., 2001)), they are not often combined into one single system that tries to add their respective advantages. This is the key point of the method presented in the following sections.

3.1. Regular grammars and robustness to ASR errors

When dealing with text input, hand-written rule-based systems have proven to give the best performance on the NE extraction task of MUC-7 (Bikel et al., 1999). But when the input is noisy (lack of punctuation or capitalization for example) or when the text is generated by an ASR system, data-driven approaches seem to outperform rule-based methods (Bikel et al., 1999). However, by carefully designing rules specific to ASR

transcripts, good performances can be achieved (Kim and Woodland, 2000).

This is also the case when the NEs to process are numerical expressions: NEs, like ID numbers, dates or phone numbers are generally expressed according to a set of fixed patterns which can be easily modeled by some hand-written rules in a regular grammar. These rules can be obtained by a study of a possibly small example corpus, manually transcribed. The main advantage of such grammars is their generalization power, regardless of the size of the original corpus they were induced from.

However, the main issue which arises when using regular grammars on automatic speech transcription data is the difficulty of taking into account recognition errors. Indeed, a simple insertion or substitution in a NE expression will lead to a rejection of the expression by the grammar. A NE detection system based only on regular grammars applied to the best string hypothesis generated by the ASR module will have a very high false rejection rate because of the numerous insertions, substitutions and deletions occurring in the ASR hypothesis.

Two possible ways of addressing this problem are as follows:

- replace the regular grammars by a stochastic model in order to estimate the probability of a given distortion of the canonical form of a NE;
- apply the regular grammars, not only to the best string hypothesis of the ASR module, but to a word-graph produced during the recognition process (or generated from the *n*-best strings generated).

These solutions are discussed in the two following sections.

3.2. Using statistical taggers

As we previously pointed out, adding probabilities to grammars is one possibility for dealing with ASR errors and spontaneous speech effects. This can be done by means of Probabilistic Context-Free Grammars (PCFG). However, this method implies writing grammars that model all the possible distortions that might occur and estimate the likelihood of each of them, which is not an easy task. Because of this difficulty and with regards to the poorer results obtained by Huang et al. (2001) with this method compared to those they obtained with a rule-based system or a Maximum-Entropy tagger, we decided to implement a simpler model based on a tagging approach. In this case, all the possible strings are accepted by the grammar and the scores attached to each derivation are estimated on a training corpus.

Tagging methods have been widely used in order to associate with each word of a text a morphological tag called part-of-speech (POS). In the framework of NE detection, we consider one tag for each kind of NE and a default tag corresponding to the background text, between each NE expression. Two kinds of models have been proposed for dealing with NE detection: one is based on a state-dependent Language Model approach considering the transition probabilities between words and tags within a sentence (Bikel et al., 1999; Palmer et al., 1999). The other one is based on a Maximum-Entropy (ME) model (Borthwick et al., 1998). Both approaches heavily rely on the features selected for estimating the probabilities of the different models. We chose to implement a tagging approach based on a Language Model (LM), very close to the standard LMs used during the speech recognition process.

However, the trade-off with this kind of method is the amount of labeled data needed in order to train the models. This point is particularly crucial in a spoken dialogue context where, as we previously said, the NE tag set is dependent on the application and where the size of the training corpus available for a given application is often rather small.

3.3. Parsing a word lattice

Several methods have been proposed in order to apply parsing techniques to word graphs. Some of these methods (Chappelier et al., 1999; Kieffer et al., 2000) aim to efficiently produce parse trees with chart based parsing algorithms, without rescoring the arcs of the graphs. Other methods use grammars in order to calculate the best path in the graph, by mean of a $A*$ algorithm (Chelba and

Jelinek, 2000) or a Markov parsing technique (Roark, 2002). In our case, because the grammars we use are regular grammars, they can easily be represented by Finite State Machines (FSM). Applying such FSMs to word graphs is a straightforward operation if the graphs are also represented by FSMs (see Mohri et al., 2000, 2002 for more details about regular grammars and FSM).

However, because of data sparseness problems, the grammars we use in order to represent the NEs are not stochastic ones. Therefore, applying such grammars to word graphs can only provide parse trees and answer the question: Which NEs can be contained in a given word graph? The only scores attached to the different paths in the graphs are those obtained through the acoustic and language models from the ASR component. We show in the next section how this method can be associated with the NE tagger previously presented in order to overcome this problem.

3.4. A hybrid approach

The approach proposed is based on a 2-step process, which tries to take advantage of the two methods previously presented: firstly, we detect NEs on the 1-best hypothesis by means of the NE tagger; secondly, once we have detected areas in the speech input which are likely to contain NEs, we verify that we can find a match for each of them in the word lattice with the regular grammars representing the NEs. These grammars are applied *only* on the areas selected by the NE tagger.

The general idea is, when processing an utterance, to use the tagger in order to have a general idea of its content, then to go back to the word-lattice and refine the transcription with very constrained models (the regular grammars) applied locally to the areas detected by the tagger. By doing so we link together the understanding and the transcribing processes and the final transcription output is a product of the Spoken Language Understanding module instead of the ASR module. Fig. 1 presents the general architecture of a Spoken Dialogue system following this strategy. All the different modules presented in this picture are going to be detailed in the following sections.

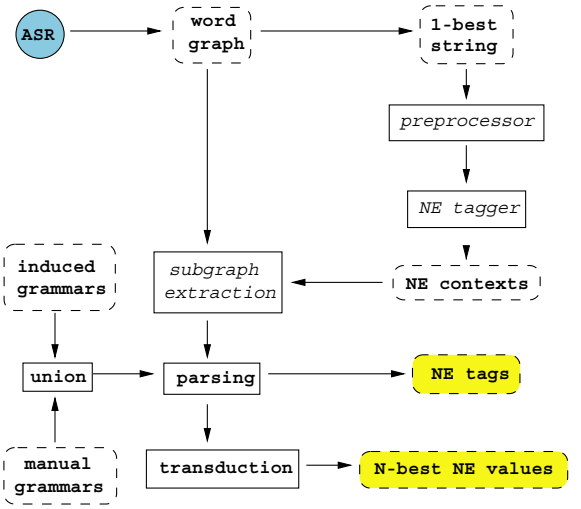

Fig. 1. General architecture of the NE module.

This hybrid method presents several advantages compared to each method taken separately:

- The Language Model used by the tagger can afford to be poorly trained because of a lack of training corpus. Indeed, because the second step of the process verifies the occurrence of the NEs that are detected, the tagger can over generate NE tags.
- Using non-stochastic grammars is not a problem here because the aim of these grammars is not to help find the best path in the graph but to check the existence of a parse tree, representative of a given NE, in an area detected by the tagger.

Before presenting a comparison of the results obtained with these various methods, the following sections highlight some of the key points of the method which is proposed.

4. Taking into account recognition errors in the statistical NE tagger

Part of the originality of this work is an attempt to explicitly model the ASR errors in the NE tagger Language Model. Before presenting this feature, the probabilistic model and the training

corpus of the tagger are briefly described in the next sections.

4.1. Probabilistic model

Following the formal presentation of tagging models in (Charniak et al., 1993), we can define NE detection as finding the best sequence of tags $t_{1,n}$ over a string of n words $w_{1,n}$. Let us denote this sequence as $\tau(w_{1,n})$. Each tag corresponds either to a NE tag or to the background text tag. $\tau(w_{1,n})$ is calculated by Eq. (1).

$$\tau(w_{1,n}) = \arg\max_{t_{1,n}} P(t_{1,n}, w_{1,n}) \tag{1}$$

By suitably defining terms like $t_{1,0}$ and their probabilities, we obtain Eq. (2).

$$P(t_{1,n}, w_{1,n}) = \prod_{i=1}^{n} P(t_i|t_{1,i-1}, w_{1,i-1}) P(w_i|t_{1,i}, w_{1,i-1}) \tag{2}$$

In order to collect these probabilities we make the following Markov assumptions:

$$P(t_i|t_{1,i-1}, w_{1,i-1}) = P(t_i|t_{i-2,i-1}, w_{i-2,i-1}) \tag{3}$$

$$P(w_i|t_{1,i}, w_{1,i-1}) = P(w_i|t_{i-2,i}, w_{i-2,i-1}) \tag{4}$$

In other words, we assume that the tag t_i is only dependent on the two previous words and tags. Similarly, the word w_i is dependent on the two previous words and tags as well as the knowledge of its tag. Unlike the usual POS tagging methods, we do not assume that the current tag is independent of the previous words. This assumption is usually made because of the data sparseness problem, but in our case, we can afford to integrate the words into the history as firstly the number of tags is limited and secondly the number of different words which can be part of a NE expression is also very limited (usually digits, natural numbers, ordinal number, and a few key words like: dollars, cents, month name, ...). With these assumptions, we get the following equation:

$$\tau(w_{1,n}) = \arg\max_{t_{1,n}} \prod_{i=1}^{n} P(t_i|t_{i-2,i-1}, w_{i-2,i-1}) \times P(w_i|t_{i-2,i}, w_{i-2,i-1}) \tag{5}$$

In order to estimate the parameters of our model, we need to build a training corpus, which is presented in the following section.

4.2. NE training corpus

The probabilities of our tagging model are estimated on a training corpus, which contains human-computer dialogues manually transcribed. Each NE of class `N` occurring in an utterance is located by inserting a tag `<N>` at the beginning of the NE and a tag `</N>` at the end of it. The meaning of the *beginning* and the *end* of a NE must be specified in an annotator guideline document specific to the application. As was said in Section 2, all the NE tags used in HMIHY are dialogue-dependent, associating semantic information with the NE tags. For this reason, the context of occurrence of a given NE has been defined as the smallest portion of an utterance containing the NE value as well as the context justifying the attribution of the tag.

For example, if an utterance contains two dates, one related to the bill, the other one related to a phone call, the corresponding NEs are `Which_Bill` and `Item_Date` and their contexts must contain enough information for deciding which is which. This is illustrated by the following utterance:

```
on my <Which_Bill> bill dated November 12th </Which_Bill> there's a
<Item_Date> call on September 8th </Item_Date> to Sarasota for
<Item_Amount> 12 dollars </Item_Amount> I don't recognize
```

When dealing with general NEs, like any dates, these contexts are restricted to the NE values (`November 12th` and `September 8th` on the previous example). Then, to each word w_i of this corpus is attributed a tag t_i where $t_i = t^0$ if the word does not belong to any NE expression, and $t_i = t^n$ if the word is part of an expression corresponding to the NE tag n.

The last step in the training corpus process is a non-terminal substitution process applied to digit strings, ordinal numbers, month and day names. This process increases the generalization power of our tagger by replacing some words by general

non-terminal symbols. This is especially important for digit strings, as the length of a string is a very strong indicator of its purpose. For example, 10-digit strings are very likely to represent phone numbers, but if all the digits are represented as single tokens in the training corpus, the 3-gram LM used by the tagger would not be able to model accurately this phenomenon as the span of such a model is only three words.

In contrast, by replacing the 10-digit string in the corpus by the symbol `$digit10`, a 3-gram LM will be able to correctly model the context surrounding these phone numbers. According to that consideration, all the *n*-digit strings are replaced by the symbol `$digit` *n*, the ordinal numbers are replaced by `$ord`, the month names by `$month` and the day names by `$day`.

With these treatments, and by considering that the tag `Item_amount` is labeled 1, the tag `Item_Date` is labeled 2 and the tag `Which_Bill` is labeled 3, the previous example becomes:

```
on_0 my_0 bill_3 dated_3 $month_3
$ord_3 there's_0 a_0 call_2
on_2 $month_2 $ord_2 to_0 Sarasota_0
for_0 $digit2_1 dollars_1
I_0 don't_0 recognize_0
```

The parameters of the probabilistic model of our tagger are then directly estimated from this corpus by means of a simple 3-gram approach with back-off for unseen events.

4.3. Modeling the ASR system behavior

Increasing the robustness of extraction information systems to ASR errors is one of the current big issues of Spoken Language processing. Even if statistical models are much more robust to ASR errors than rule-based systems, the models are usually trained on manually transcribed speech and the ASR errors are not taken into account explicitly. This strategy certainly emphasizes the precision of the detection, but a great loss in recall can occur by not modeling the ASR behavior. For example, a word can be considered as very salient information for detecting a particular NE tag. But if this word is, for any reason, very often badly recognized by the ASR system, its salience would not be useful to the ASR output.

Several studies have tried to increase the robustness of their models to ASR errors, first by randomly generating errors in order to introduce noise into the training data (Palmer et al., 1999; Grishman, 1998), then by using errors made by an ASR system (Palmer, 2001). The latter approach can be considered as one of the first successful attempt to model ASR errors in automatic speech transcription processing. One very interesting result shown in (Palmer, 2001) is that it is crucial to train the models with *real* errors made by the same system as the one which is going to be used for the tests. Generating errors, randomly or based on speech science consideration, does not bring any improvement.

In our method, the whole training corpus is processed by the ASR system in order to learn automatically the confusions and the mistakes that are likely to occur in the deployed system. This ASR output corpus is then aligned, at the word level, with the transcription corpus. A symbol `NULL` is added to the ASR transcript for every deletion and each insertion is attached to the previous word with the symbol `+`. By this means, both manual transcriptions and ASR outputs contain the same number of tokens. The last process consists simply of transferring the tags attached to each word of the manual transcription, as presented in the previous section, to the corresponding token in the ASR output.

Such a method balances the inconvenience of training a model directly on a very noisy channel (ASR output) by structuring the noisy data according to constraints obtained on the clean channel (manual transcriptions).

With this method the recognition errors are not explicitly detected because they are integrated in the Language Model at the same level as the correct words. However, if a NE tag is detected by the tagger on an incorrect string of words, it is the second step process that is in charge of correcting the ASR errors. This is done by finding, in the area of the word graph selected by the tagger, the best string of words accepted by a regular grammar representing the NE tag detected. The comparison between training on manual transcriptions and aligned ASR output is presented in Section 8.

5. Dealing with dialogue-dependent named-entities

The NE tagging process consists of maximizing the probability expressed by Eq. (5) by means of a search algorithm. The input is the best-hypothesis word string, output of the ASR module, and pre-processed in order to replace some tokens by non-terminal symbols as in Section 4.2. We chose to apply the tagger to the 1-best path instead of the whole word graph for two reasons: firstly, the tagger is trained in order to maximize the recall, as will be presented in Section 8.1, and because of this biased training this model is not suited for choosing the best word string as well as the best sequence of tags; secondly, this saves the computational cost of applying another Language Model to each transition of the word graph.

As we previously said, most of the NEs used in spoken dialogue contexts carry information about their functions within the dialogue. For the latter, the size of the word context needed in order to represent them can be quite long, and in most cases much longer than the usual span of traditional Language Models, which is only three words (3-gram models).

For this reason, it is interesting to use a model which can analyze the whole context of occurrences that are detected and calculate a score for the probability of a given context to represent a specific dialogue-dependent NE tag. Following what is usually done in the Spoken Language Understanding module for detecting the call type of an utterance (Tur et al., 2002), we decided to implement a text classification approach for the evaluation of the detected contexts.

This text classifier, which allows us also to tune the precision and the recall of our model, is trained as follows:

(1) the ASR output of the training corpus is processed by the NE tagger;
(2) on one hand, all the contexts detected and correctly tagged according to the manual labels are kept and marked with the corresponding NE tag;
(3) on the other hand, all the false positive detections are labeled with the tag `OTHER`;
(4) finally the text classifier is trained in order to separate these samples according to their NE tags as well as the `OTHER` tag.

During the tagging process, the scores given by the text classifier are used as confidence scores to accept or reject a NE tag according to a given threshold. The text classifier used in the experimental section is a decision-tree classifier based on the semantic-classification-trees introduced for the ATIS task by Kuhn and Mori (1995) and used for semantic disambiguation in (Béchet et al., 2000). This classifier accepts multiple class samples.

6. Merging knowledge-based and data-induced methods

As we previously said, most of the relevant NEs that can be found in a spoken dialogue context follow some fixed patterns that can be easily modeled by hand-written regular grammars. However, disfluencies (e.g., "uh", "um", repeated words, self-repairs) are prevalent in the spontaneous utterances of normal speakers and it is important to adapt the grammars in order to handle them.

Integrating disfluencies in hand-written grammars is not an easy task, as it is hard to predict a priori where they are going to occur. That is why it seems interesting to directly induce such grammars from corpora containing spontaneous utterances manually transcribed.

6.1. Automatically inducing grammar

Several approaches have been proposed for inducing grammars from text corpora (Ron et al., 1998; Carrasco and Oncina, 1999; Stolcke and Omohundro, 1994). For example, Stolcke and Omohundro (1994) propose a Bayesian model merging approach for inducing probabilistic grammars. Adding a new sample string to a Context-Free Grammar with a start non-terminal S consists of simply adding a new top-level production rule (for S) that covers the sample precisely. Then, a non-terminal is added for each new

terminal of the right-hand side of the new rule in order to facilitate the merging process.

The key point of all the grammar induction methods is the strategy chosen for merging the different non-terminals: if no merging is performed, the grammar will only model the training examples, if too many non-terminals are merged, the grammar will accept incoherent strings. In our case, because the word strings representing the NE contexts are already quite small, and because the induction of a wrong pattern can heavily affect the performance of the system by generating a lot of false positive matches, we decided to limit the merging strategy to a set of standard non-terminals: digit, natural numbers, day and month names. The following substitutions are considered:

- each digit (0–9) is replaced by the token `$digitA`;
- the natural numbers from 10–19 are replaced by the token `$digitB`;
- each multiple of ten, except 10, (20, 30, 40, ..., 90) is replaced by the token `$digitC`;
- each ordinal number is replaced by the token `$ord`;
- each name representing a day is replaced by the token `$day`;
- each name representing a month is replaced by the token `$month`;

For example, the six following contexts, corresponding to the tag `Item_Amount`

– `charged for two ninety five;`
– `charging me a dollar sixty five;`
– `charged a dollar sixty;`
– `charges of thirty dollars;`
– `charge of eleven dollars and sixty three cents;`
– `charged to me for four dollars and forty eight cents;`

become:

– `charged for $digitA $digitC $digitA;`
– `charging me a dollar $digitC $digitA;`
– `charged a dollar $digitC;`
– `charges of $digitC dollars;`
– `charge of $digitB dollars and $digitC $digitA cents;`
– `charged to me for $digitA dollars and $digitC $digitA cents;`

The symbols `$digitA`, `B`, `C`, `$ord`, `$month` and `$day` are considered, here, as terminal symbols. The same kind of preprocessing operation will be performed on the text-strings that are going to be parsed by the grammars. Despite the size of the available corpus, we do not have enough data for learning a reliable probability for a given NE to be expressed in a given way. We therefore consider that all rules are equal and the grammars obtained are not stochastic.

Each grammar rule obtained for a given NE tag is turned into a Finite State Machine (FSM), and the complete grammar for the tag is the union of all the different FSMs extracted from the corpus. Fig. 2 illustrates this process by representing the FSM corresponding to the six `Item_Amount` contexts previously given as examples.

6.2. Merging hand-written and data-induced grammars

After the training phase, each NE tag is associated with an induced grammar modeling its different expressions in the training corpus. It is therefore possible to enrich such grammars with hand-written ones. The only constraint here is to write grammars accepting regular languages. This constraint is due to the representation and the parsing strategy chosen: because we use FSMs and composition operation between them, only regular languages can be directly represented this way. However, by using methods like the one proposed in (Mohri and Nederhof, 2000), it is possible to approximate any context-free language with regular languages.

Because none of the data-induced or hand-written grammars are stochastic, their merging process is straightforward: the merged grammar is simply the union of the different FSMs corresponding to the different grammars. These hand-written grammars can be seen as a back-off strategy for detecting NEs. For example, the NE tag `Which_Bill` can have either a value corresponding to a date (the bill-issue date, for example) or to a

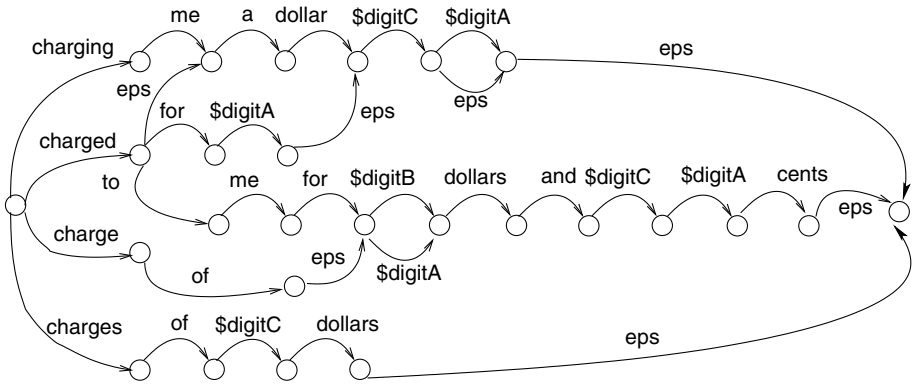

Fig. 2. FSM induced from data.

relative position (current or previous). But not all dates can be a considered as a `Which_Bill`, for example, a date corresponding to a given phone call. In the NE context training corpus, all the dates corresponding to a tag `Which_Bill` are embedded in a string clarifying the nature of the date, like: *bill issued on November 12th 2001*. Therefore all the strings matching a grammar built from this example are very likely to represent a `Which_Bill` tag. However, if the NE is expressed in a different way, which is not represented in the training corpus, like *bill dated November 12th 2001*, the grammar will reject the string. This data sparseness problem is inevitable whatever the size of the training corpus when the application is dealing with spontaneous speech.

Adding hand-written rules is then an efficient way of increasing the recall of the NE detection process. For example, in the previous example, if a hand-written grammar representing any kind of date is added to the data-induced grammar related to the tag `Which_Bill`, all the expressions identifying a bill by means of a date will be accepted. The first expression *bill issued on November 12th 2001* will still be identified by the pattern found in the training corpus, because it is the rule that gives the better coverage on the context selected by the tagger which is chosen; the second expression *bill dated November 12th 2001* will be reduced to the date itself and accepted by the back-off hand-written grammar representing the dates.

However, if this technique improves the recall, the precision can drop because of false positive detections generated by the non-context-dependent rules of the hand-written grammars.

7. Extracting an *n*-best list on the NE values

Once a NE context is detected by a grammar, a NE value has to be extracted. Each NE tag can be represented by one or several kinds of values. Unlike the NE extraction task of MUC-7 and DARPA HUB-4 programs, the evaluation of a NE processing method applied to Spoken Dialogue systems should be done on the values extracted and not the word-string itself. From the Dialogue Manager point of view, it is the normalized values of the NEs that will be the parameters of any database dip, and not the string of words used to express them. For example, the value of the following NE *bill issued on November 12th 2001* is *2001/11/12*, and if the same value is extracted from the ASR output, this will be considered as a success, even if the NE string estimated is *bill of the November 12th 2001* or *issue the November 12th of 2001*. Evaluating the values instead of the strings is called here the evaluation of the understanding accuracy.

Extracting a value from a word-string is not always straightforward: some ambiguities exist, even for standard NEs like phone numbers. For

example, the following number *220 386 1200* can be read as *two twenty three eight six twelve hundred* and this string can then be turned into these following digit strings:

```
2203861200 223861200 22038612100
2238612100
```

In order to produce correct values, we implemented a transduction process that outputs values during the parsing step by means of the NE grammars already presented. The result of this transduction on the previous phone string will be:

```
two->2 twenty->20
three->3 eight->8 six->6
twelve->12 hundred->00
```

For the hand-written grammars, this is done by simply adding to each terminal symbol the format of the output token that has to be generated. For example, the previous transduction is made by the rule:

```
<PHONE> ->
  $digitA/$digit1  $digitC/$digit2
  $digitA/$digit1  $digitA/$digit1
                   $digitA/$digit1
  $digitB/$digit2 hundred/00
```

with `$digit1` corresponding to the first digit of the input symbol, `$digit2` to the first two digits of the input symbol, and `00` to the digit string 00.

The same process is used for data-induced grammars. In this case, we first align, word to word, the word context and the value of each sample manually obtained on the training corpus. The symbols that do not produce any output token are transduced into the *epsilon* symbol, and similarly the output tokens that are not produced by a word from the NE context are considered emitted by the same *epsilon* symbol. This alignment is done by means of simple rules that make the correspondence, at the word level, between input symbols and output tokens.

Extracting a value from a word graph consists of a transduction process between this word graph turned into an FSM and the grammar representing the FSM. On the grammar side, we simply transform the FSMs into transducers by adding the output tokens attached to each input symbol for each arc of the FSMs. On the ASR output side, we perform the following process:

(1) if the ASR output is a 1-best word string, we turn it into a sequential FSM, otherwise we use the word graph directly as an FSM;
(2) the FSM obtained is turned into a transducer by duplicating each word attached to each arc as an input and output symbol;
(3) each output symbol belonging to one of these non-terminal classes: `$digitB`, `$digitC`, `$ord`, `$month` and `$day`, is replaced by the name of its class.

The extraction process is now a by-product of the detection phase: once a string is accepted by a grammar by means of a composition operation between their corresponding transducers, at the same time, the matching of the input and output symbols of both transducers removes any ambiguities for the translation of a word string into a value. To obtain a value, we just have to choose one path in the composed FSM, filter all the epsilon output symbols and then match the other input–output symbols.

Such a process is illustrated in Fig. 3. *FSM1* is the transducer corresponding to the ASR output; *FSM2* is one of the grammars automatically induced from the training data, which represents the transduction between a NE context like *twenty two sixteen charge* and the value *22.16*. From *FSM1*, the following values can be extracted:

```
64.10 64.00 60.10 60.00 60.04 4.10
4.00 10.00 0.64 0.60 0.04 0.10
```

But after the composition between the two FSMs, the following transduction occurs:

```
sixty->$digit1   four->$digit1
eps->.
ten->$digit2   charge->eps.
```

This means that, in order to produce a value, we have to take the first digit of *sixty*, the first digit of *four*, then we have to add the token '.', take the first two digits of *ten* and finally erase the word *charge*. We obtain, from the twelve possible values previously enumerated, only one match, which is 64.10.

One of the main advantages of this approach is the possibility of generating an *n*-best solution *on the NE values* instead of the NE strings. Indeed, each path in the composed FSM between the ASR

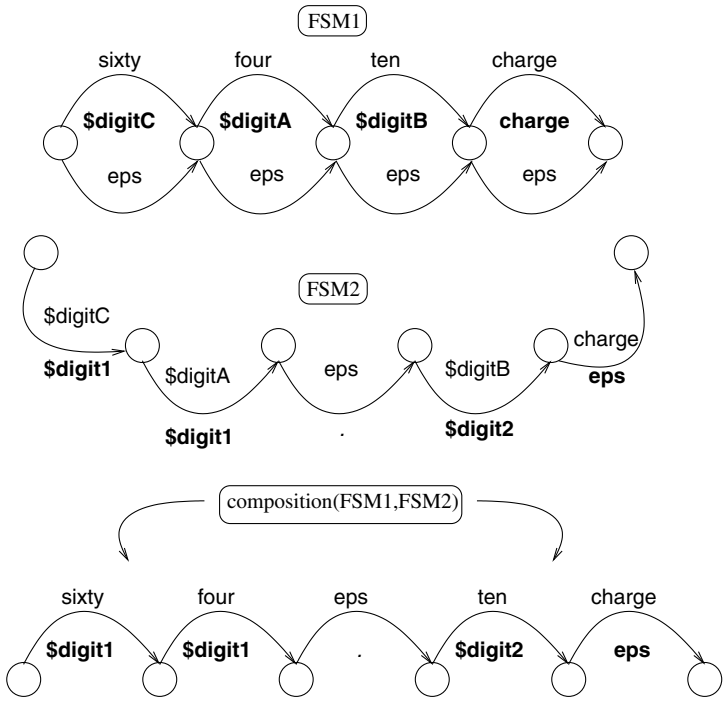

Fig. 3. NE value transduction.

output transducer and the grammar transducer (once all the epsilon transitions have been removed) corresponds to a different NE value.

Enumerating the *n*-best paths leads to enumerating the *n*-best values for a given NE tag on a given context. In contrast, if the *n*-best generation is done on the word lattice alone, one has to generate a much bigger set of paths in order to obtain different values, as most of the *n*-best paths will differ only by words that are not relevant for extracting a value.

Extracting the *n*-best values can be extremely useful in a dialogue context, as some extra information (customer data, constraints on the values, etc.) can be used to select a value among a list of hypotheses. For example, Rahim et al. (2001) shows that using a general phone directory for checking the relevance of phone numbers extracted is a very efficient filter: the understanding accuracy of the phone strings that belong to the directory is 94.5% (and this represents 61% of the hypotheses) compared to only 45% accuracy for those that cannot be found in any phone directory.

8. Experiments

8.1. Experimental setup

The HMIHY corpus used in this study contains a 102K utterance training corpus and a 28K utterance test corpus, all from live customer traffic. 15% of these utterances contain at least one NE occurrence. The Language Model used by the tagger is trained only on this 15% of the corpus. This leads the tagger to over-generate NE tags in order to have the highest recall possible. The low precision that results from this training process is corrected by the NE grammars, as we will see in the following results. The grammars used for NE extraction consist of about 2K rules automatically induced from the training corpus together with a set of 94 hand-written back-off

rules. We compare three different methods in order to evaluate all the different techniques previously presented:

(1) **Gram** corresponds to the grammar method alone presented in Section 3.1 applied to only the 1-best string of the ASR output;
(2) **Tagger** corresponds to the tagger alone applied to the 1-best string;
(3) **Hybrid** corresponds to the hybrid method presented in Section 3.4, applied to the whole word graph (coded as an FSM) output by the ASR module.

In order to evaluate the method proposed for dealing with ASR errors, the results are presented according to the kind of corpus used to train the tagger: manual transcriptions (*clean*) or ASR outputs (*noisy*).

We give the results separately for the detection task and the extraction task. In the detection task, the system answers the question: *Does this utterance contain a phone number or a date?*; in the extraction task we measure the understanding accuracy of our methods, which means answering the question: *If this utterance contains a phone number or a date, what are their values?* The results are given according to the standard following measures: Precision P, Recall R and F-measure F. They are defined as follows:

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

$$R = \frac{\#\text{ of correct answers}}{\#\text{ of reference tokens}} \quad F = \frac{2 * R * P}{R + P}$$

Two sets of experiments have been defined:

- The first set concerns the evaluation of two *generic* tags `phone` and `date`. The aim is to compare the grammar and the tagger approaches on a simple task without taking into account the semantic ambiguities of the other application-specific NE tags.
- The second set of experiments concerns *real* dialogue-dependent NE tags and the aim is to measure the performance of our methods for two ambiguous application-specific tags.

8.2. Generic tags: phone and date

These two tags are defined in the following way:

- `phone`: any string of words which can be parsed by the phone grammar used in the system (containing data induced and hand written rules) is considered as a NE phone number.
- `date`: any date expression that can be parsed by a hand written date grammar and which contains at least a month and the day of a month is considered as a NE date.

8.2.1. Preliminary analysis of the task

A study we made on a set of dates expressed on a subset of the HMIHY training corpus showed us that 82.3% of them followed the previous date grammar. The remaining dates were expressed with a cardinal number instead of an ordinal number for 6.5% of them, and only with digits, like 06/10, for 11.1%. Even if it seems incorrect to reject dates expressed with a cardinal number (e.g. *September five*), we decided to do so because the tag `date` is not part of the HMIHY tag set, and therefore is not part of the manually labeled data. Because we had to use an automatic process to obtain our reference corpus on the transcribed dialogues, we chose a set of very restrictive definitions which have the advantage of being unambiguous.

Similarly, a study of phone number expressions in the same corpus showed that 90% of them followed our grammar. The 10% remaining corresponds to phone numbers expressed with more than eleven digits or less than 10. This can correspond to some errors or repairs, but in any case we cannot extract an exact value for each of them only from the manual transcription and they are rejected (of course, going back to the speech signal would allow us to obtain a phone number value for some of them).

The test corpus we used consists of 26.7K dialogue turns from the test corpus of HMIHY corresponding to real dialogues of the deployed system between November 2000 and January 2002. We parsed the manual transcription of this corpus with our date and phone grammars in order to constitute our reference corpus. 464 dates and

1256 phone numbers tags and values have been found.

For the tag phone number, we consider two conditions: the first one contains the utterances following the system prompt: *Please give me your home phone number starting with the area code*; the other set of utterances groups together responses to all the other prompts. The first condition is called `phone(1)` and contains 617 utterances with 519 phone numbers; the other one is called `phone(2)` and contains 26.1K utterances with 737 phone numbers.

These two conditions correspond to two tasks of very different perplexity. In `phone(1)`, the utterances are very likely to contain a phone number alone, as it is requested in a direct question. Indeed, the average length of the utterances containing a phone number is 10.9 words. In `phone(2)` the phone numbers are not expected and they are usually embedded in long utterances along with various other digit expressions. The average length of the utterances containing a phone number is 39.2 words.

8.2.2. Detection results

Table 2 presents the detection results according to the different methods on the 3 test sets. As expected, on one hand the grammar based method obtains the best precision scores over all the other methods but also the poorest recall scores. On the other hand, because the tagger is trained in order to maximize the recall measure, the NE tagger alone obtains the best recall scores but the poorest precision ones. By combining both methods and by means of word graphs instead of 1-best strings, the hybrid method combines the advantages of both methods and obtains the best F-measure scores on `date` and `phone(2)`. The tagger alone is slightly better on `phone(1)` because these occurrences correspond to a system-initiative dialogue step: the prompt asks directly for a phone number, so as soon as a digit-string is recognized, it is very likely to correspond to a phone number.

Modeling the ASR output explicitly on the training of the tagger increases the recall measure. However a decrease in the precision measure of the detection is noticeable. These false positive detections correspond mostly to phone numbers expressed by the caller with an erroneous digit string (and in consequence not tagged as phone numbers in the reference corpus), or to dates expressed with a cardinal number instead of an ordinal number (6.5% of the dates according to our training corpus). Indeed, some ordinal numbers are very easily confused, from an acoustic point of view, with their corresponding cardinal numbers (*fourth* and *four* for example). That is why, even if the 1-best string correctly contains the cardinal number, it is very likely that its corresponding ordinal is present in the word-lattice produced by the ASR. The tagger trained on the ASR will consider this distortion as acceptable, and the grammars applied to the corresponding portion of the lattice will use the inserted ordinal and generate a false positive.

Table 2
Detection results with grammars, tagger and hybrid method according to the kind of corpus used to train the tagger: *clean* means a training on the manual transcriptions of the training corpus; *noisy* means a training on the ASR output of the training corpus

	Gram	Tagger		Hybrid	
		Clean	Noisy	Clean	Noisy
`date`					
P	97.1	39.3	23.9	87.2	85.9
R	43.8	72.6	82.5	57.1	65.7
F	60.3	51.0	37.1	69.0	74.5
`phone(2)`					
P	98.5	52.9	42.3	96.3	94.8
R	65.7	78.8	93.9	71.8	80.9
F	78.9	63.3	58.4	82.3	87.3
`phone(1)`					
P	99.8	98.2	94.7	99.8	99.3
R	76.9	84.0	96.7	79.8	89.0
F	86.6	90.6	95.7	88.7	93.9

Fortunately, if this decrease in the precision measure is significant when the tagger is used alone (15.4% loss for `date` and 10.6% loss for `phone(2)`), this degradation is much smaller in the hybrid method (1.3% for `date` and 1.5% for `phone(2)`) while the gain in recall remains significant (8.6% gain for `date` and 9.1% gain for `phone(2)`).

8.2.3. Extraction results

Table 3 presents the results according to the understanding accuracy measure. A NE detected is

Table 3
Understanding accuracy of grammar and hybrid methods according to the corpus used to train the tagger (clean = manual transcription, noisy = ASR output)

	Gram	Hybrid	
		Clean	Noisy
`date`			
P	87.6	75.0	73.2
R	39.4	49.1	56.0
F	54.4	59.4	63.5
`phone(2)`			
P	88.4	80.1	74.1
R	59.0	59.8	63.3
F	70.8	68.5	68.3
`phone(l)`			
P	91.9	89.5	86.7
R	70.8	71.6	77.7
F	80.0	79.6	81.9

Table 4
Understanding accuracy of the hybrid method (trained on the ASR corpus) for the n-best values produced by the extraction process ($n = 1, 2, 5, 10$)

	1-best	2-best	5-best	10-best
`date`				
P	73.2	78.0	79.2	79.4
R	56.0	59.7	60.6	60.8
F	63.5	67.6	68.6	68.9
`phone(2)`				
P	74.1	79.5	83.8	84.4
R	63.3	67.9	71.5	72.0
F	68.3	73.2	77.2	77.7
`phone(l)`				
P	86.7	89.5	91.3	92.4
R	77.7	80.2	81.8	82.8
F	81.9	84.6	86.3	87.3

considered a success only if both its tag and its value are correct according to the reference corpus. Because the tagger does not output NE values, the two methods compared are the grammar one and the hybrid one.

As in the previous table, the hybrid method gives better recall measures than the grammars used alone. However, this gain in recall is smaller than the one observed for the detection task while the drop in precision between the grammars alone and the hybrid method is still significant. Indeed, while the F-measure score is still better with the hybrid method for the tags `date` and `phone(l)` when the training is done on the *noisy* corpus, the grammars remain better for `phone(2)`.

This means that, although the NE tags detected are correct (according to Table 2), the values extracted are often incorrect. This can be explained by the following remark: if a correct NE is not detected by the tagger or the grammars, it means that its expression is noisy in the ASR output and it is very likely that the value extracted from it is erroneous. In this case, it is interesting to see if the performances get better by taking into account not only the best value extracted, but the n-best values as presented in Section 7. These results are presented in Table 4. In this experiment the reference values are compared not only to the best NE value extracted but to the n-best ones with $n = 1$, 2, 5 and 10.

As we can see a very significant gain can be observed by taking into account even a very small n-best list of values. For example, by simply keeping the first two values, the gain in the F-measure score for the `date` extraction is 4.1%, for `phone(2)` the gain is 4.9% and for `phone(l)` the gain is 2.7% (all these improvements are absolute percentages). Of course these are *oracle* improvements because the system does not automatically know how to choose among the best possibilities. However, in practice, *all* NE values may be passed to the Dialogue Manager for later filtering as presented in Section 7.

8.2.4. Comparison with other studies

We can also compare these results to those presented in other studies on the same kind of NE. The experiments on `phone(l)` are very similar to those presented in (Rahim et al., 2001). The utterances processed are all answers to a prompt like: *Please give me your phone number starting with the area code*. The acoustic and language models are dedicated to recognizing *numeric language* and the phone numbers are represented by hand-written rules that translate the ASR output into digit strings. The understanding accuracy obtained is 74% (corresponding here to the recall measure). This result is better than the one

obtained with the grammars alone (certainly because the ASR module is specifically tuned for processing numeric language) but is not as good as the one obtained with the hybrid method (77.7%). Using specific models in this particular context before applying our method can be a way of adding the gains of both methods.

Another comparable study is presented in (Huang et al., 2001) about the recognition of phone numbers in the Voicemail (Padmanabhan et al., 1999) corpus. This corpus contains several hours of conversational telephone speech and is very close to the type of data contained in the HMIHY corpus. The phone numbers to extract are embedded in the utterances and therefore comparable to the ones that can be found in the test `phone(2)`. Various methods are evaluated (hand-written rules, stochastic grammars, Maximum-Entropy tagger) and the best results obtained on the understanding measure of phone numbers are a precision of 56% for a recall of 52% (compared to 74.1% precision and 63.3% recall for our method). However these results are not directly comparable to ours because of differences in the difficulty of the speech tasks; e.g. the global WER in Voicemail is 35% compared to 27% in our corpus.

8.3. Dialogue-dependent tags: Item_Amount and Which_Bill

This second set of experiments aims to evaluate our methods on an application-specific task, corresponding to the detection and the extraction of two NEs from the HMIHY tag set: `Item_Amount` and `Which_Bill` which are defined in Section 2. The main difficulty for these two tags is the variable length of the context that has to be taken into account in order to decide whether or not a given expression is a NE. Having to deal with long contexts leads to a data sparseness problem, both for inducing grammars and training the tagger, as the variability of a NE expression increases with its length.

The test corpus we used for this experiment is a subset of the one presented previously. It contains 7K turns corresponding to real dialogues from the deployed system between November 2001 and January 2002. It contains 304 `Item_Amount` tags and 158 `Which_Bill` tags.

8.3.1. Detection results

Section 5 presents the method deployed for dealing with these dialogue-dependent NEs: a text classifier is added to the tagger in order to give the NE expressions detected a confidence score for representing a particular NE tag. This text classifier evaluates the NE expressions detected and their surrounding contexts. By using a threshold on the confidence scores given by the classifier, we can accept or reject a NE which is detected and therefore tune the precision and the recall of our model.

The results are given according to this threshold in Figs. 4 and 5. These curves are Receiver Operating Characteristic (ROC) curves. They represent the tradeoff between true-positive rate and false-

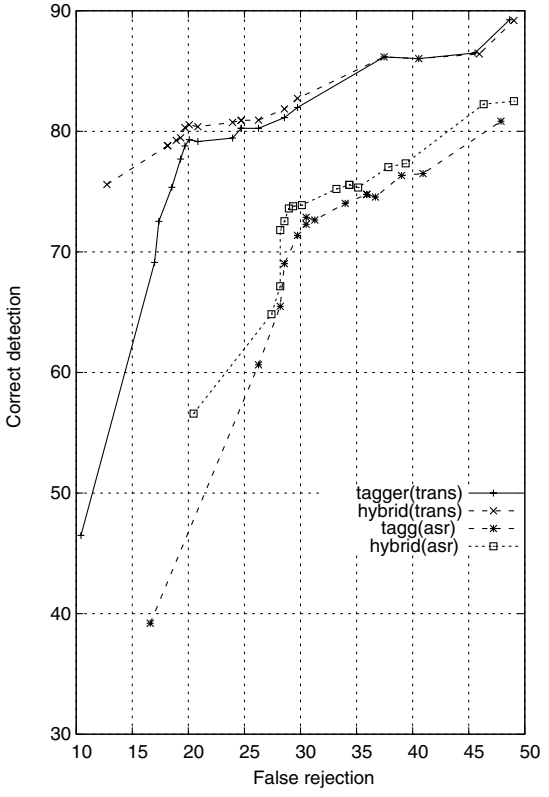

Fig. 4. ROC curve for the detection of the `Item_Amount` tag on the manual transcriptions (*trans*) and the word graph output of the ASR module (*asr*).

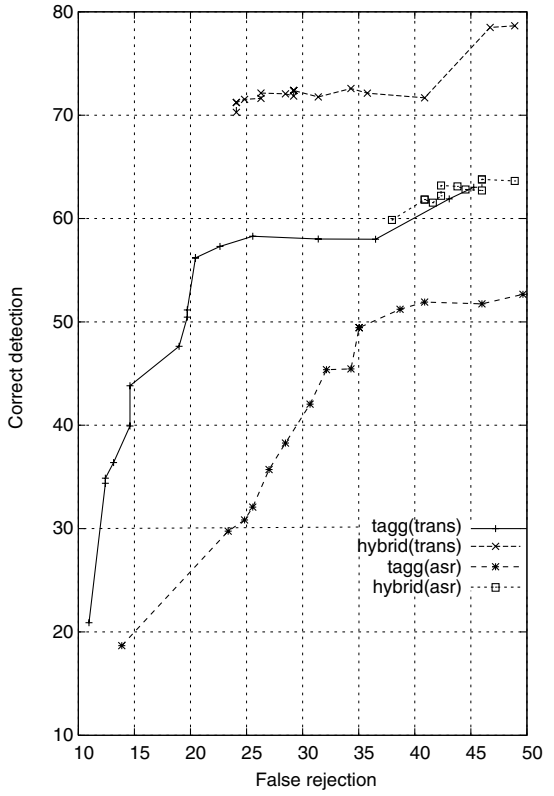

Fig. 5. ROC curve for the detection of the `Which_Bill` tag on the manual transcriptions (*trans*) and the word graph output of the ASR module (*asr*).

negative rate in binary classification problems. In these figures, the *x*-axis is the false rejection rate (*1-recall*), the *y*-axis is the correct detection (*precision*) rate. The results are given for the 2 methods: tagger + text classifier and the hybrid method (tagger + text classifier+grammars). Each method is performed first on manual transcriptions (*trans*), then on word graphs from the ASR module (*asr*). Using the manual transcriptions with the hybrid method allows us to check the coverage of the data-induced and the manual grammars used, regardless of any recognition errors.

As we can see, a significant lack of coverage is observed for the tag `Which_Bill`: for a threshold set to zero, the tagger has a false rejection rate of about 11% on manual transcriptions compared to the 24% obtained by the hybrid method. Because the only difference between these two methods is simply the checking of a matching grammar in the area detected by the tagger, this increase in false rejection is due to a lack of coverage of the `Which_Bill` expressions in the grammars induced and manually written. The same trend, although with a lower amplitude, can be observed for the tag `Item_Amount`.

But using grammars in conjunction with the tagger also leads to improving the correct detection measure for similar values of false rejection rate. This is specially true for the tag `Which_Bill` where an improvement of about 10% (absolute) in the correct detection rate is achieved, for the same values of false rejection rate, by means of grammars on the areas detected by the tagger and validated by the classifier.

8.3.2. Extraction results

The extraction results presented in Table 5 are obtained with a fixed threshold on the acceptation/rejection by the classifier. At this operating point we obtain 71.8% recall and 72.9% precision for the detection of `Item_Amount` and 59.1% recall and 61.9% precision for the detection of `Which_Bill`.

The understanding accuracy on both tags is significantly lower than the one obtained with the previous set of tags (`phone` and `date`). This highlights the intrinsic ambiguities of these semantic tags as well as the lack of coverage of the NE grammars.

The tag `Item_Amount` is particularly ambiguous. Indeed, even if the detection process gives better results on this tag than on `Which_Bill`, its

Table 5
Understanding accuracy of the hybrid method on the tags `Item_Amount` and `Which_Bill` for the *n*-best values produced by the extraction process ($n = 1, 2, 5, 10$)

	1-best	2-best	5-best	10-best
`Item_Amount`				
P	48.2	55.7	60.0	60.4
R	47.5	54.8	59.1	59.5
F	47.9	55.3	59.5	59.9
`Which_Bill`				
P	59.5	59.5	59.5	59.5
R	56.8	56.8	56.8	56.8
F	58.1	58.1	58.1	58.1

F-measure is significantly lower (47.9% vs. 58.1%). This is due to the lack of a fixed pattern for expressing a monetary value, unlike phone numbers that have to be expressed by a 10 digit string. Because very few constraints can be used in order to extract a value for a given `Item_Amount` tag, the extracted value relies mainly on acoustic evidence and this generates a lot of errors. That is why it is important to give to the Dialogue Manager not only the most probable value but the n-best values, which can be further filtered according to the dialogue context.

Table 5 shows the results obtained with 1-best, 2-best, 5-best and 10-best lists. As we can see, the gain in F-measure for `Item_Amount` is an absolute 12% by taking into account the 5-best values instead of only the first one. It is interesting to notice that, in contrast, no gain is obtained for the tag `Which_Bill` by using n-best lists. Indeed, the extraction results are very similar to the detection ones: 59.5% vs. 61.9% for the precision and 56.8% vs. 59.1% for the recall. It means that once a `Which_Bill` tag is detected, the value extracted is often correct and no improvement is observed by looking for other values in the word graph.

9. Conclusion

The method proposed in this paper attempts to bridge the gap between the Automatic Speech Recognition module and the Spoken Language Understanding module by linking more closely the understanding and transcription processes. The output of the ASR module is no longer a single string of words supposed to be perfect, but a word-graph containing a lot of different paths leading to different interpretations. It is the understanding module that chooses among all these possible interpretations in order to output the best string of words corresponding to the interpretation chosen.

Applying such a model to the NE detection and extraction tasks leads us to build a 2-step process: the *understanding* process which consists of a statistical tagger and a text classifier that select in the word graphs the areas likely to contain NE expression; the transcription process performed by regular grammars that output normalized values for each entity detected. This approach efficiently combines knowledge-based methods, like handwritten grammars, and data-driven approaches like automatically induced grammars and statistical tagging systems.

The results obtained validate this approach for the detection task: although the grammar method alone obtains the best precision scores and the tagger alone the best recall scores, the hybrid method obtains the best F-measure scores on nearly all the tests (except the very specific test on `phone(1)`) with a very significant improvement over the other methods (14.2% absolute improvement for the tag `date` and 8.4% absolute improvement for `phone(2)`).

One of the key points of this method is also the possibility of generating n-best lists *on the NE values* instead of traditional n-best lists on word strings. These values can be further filtered by the Dialogue Manager according to the dialogue context. By producing the 5-best values instead of just the first one, we obtain an improvement in the F-measure of 5 to 12% absolute (depending on the tag, except for the `Which_Bill` tag).

Integrating the ASR behavior in the training of the statistical models is also one of the more novel features of this work. The ASR errors are part of the training corpus and therefore modeled. This noise is handled by structuring the ASR output according to constraints obtained on the *clean* manually transcribed version of the corpus. This feature increases the recall of the tagger and the hybrid method leading also to a significant gain in the F-measure in most of the cases.

Finally, this model is also able to process the dialogue-dependent NEs that contain the role of the entities within the dialogue. Even if the results obtained with such tags are still quite low, mainly due to data sparseness problems, we believe that the general method proposed here can be an efficient answer for dealing with such entities.

References

Béchet, F., Nasr, A., Genet, F., 2000. Tagging unknown proper names using decision trees. In: 38th Annual Meeting Assoc. Computat. Linguistics. Hong-Kong, China, pp. 77–84.

Bikel, D.M., Schwartz, R., Weischedel, R., 1999. An algorithm that learns what's in a name. Mach. Learning: Special Issue on Natural Language Learning 34 (1–3), 211–231.

Black, W., Rinaldi, F., Mowatt, D., 1998. Facile: Description of the NE system used for MUC-7. URL: citeseer.nj.nec.com/black98facile.html.

Borthwick, A., Sterling, J., Agichtein, E., Grishman, R., 1998. Nyu: Description of the mene named entity system as used in MUC. In: Proc. Seventh Message Understanding Conf. (MUC-7). URL: citeseer.nj.nec.com/borthwick98nyu.html.

Carrasco, R.C., Oncina, J., 1999. Learning deterministic regular grammars from stochastic samples in polynomial time. RAIRO (Theoret. Informat. Appl.) 33 (1), 1-20. URL: citeseer.nj.nec.com/article/carrasco991earning.html.

Chappelier, J., Rajman, M., Aragues, R., Rozenknop, A., 1999. Lattice parsing for speech recognition. In: Proc. 6th conference on Traitement Automatique du Langage Naturel TALN'99. Cargese, Corsica, France.

Charniak, E., Hendrickson, C., Jacobson, N., Perkowitz, M., 1993. Equations for part-of-speech tagging. In: 11th National Conf. Artificial Intell. pp. 784–789.

Chelba, C., Jelinek, F., 2000. Structured language modeling. Comput. Speech Language 14 (4), 283–332.

Chinchor, N., Robinson, P., 1998. Muc-7 named entity task definition. In: Proc. Seventh Message Understanding Conf. URL: http://www.itl.nist.gov/iad/894.02/related/projects/muc/.

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

Grishman, R., 1998. Information extraction and speech recognition. In: Proc. DARPA Broadcast News Transcription and Understanding Workshop. URL: http://www.nist.gov/speech/publications/darpa98/pdf/sdrlO.pdf.

Huang, J., Zweig, G., Padmanabhan, M., 2001. Information extraction from voice-mail. In: Proc. 39th Annual Meeting Assoc. Computat. Linguistic. Toulouse, France, pp. 290–297.

Kieffer, B., Krieger, H.-U., Nederhof, M.-J., 2000. Efficient and robust parsing of word graphs. In: Wahlster, W. (Ed.), Verbmobil: Foundations of Speech-to-Speech Translation. pp. 280–295.

Kim, J., Woodland, P., 2000. A rule-based named entity recognition system for speech input. In: Proc. ICSLP'2000. Beijing, China.

Kubala, F., Schwartz, R., Stone, R., Weischedel, R., 1998. Named entity extraction from speech. In: Proc. DARPA Broadcast News Workshop.

Kuhn, R., Mori, R.D., 1995. The application of semantic classification trees to natural language understanding. IEEE Trans. Pattern Anal. Mach. Intell. 17, 449–460.

Mohri, M., Nederhof, M.-J., 2000. Regular approximation of context-free grammars through transformation. In: Junqua, J.C., van Noord, G. (Eds.), Robustness in Language and Speech Technology. pp. 251–261.

Mohri, M., Pereira, F., Riley, M., 2000. The design principles of a weighted finite-state transducer library. Theoret. Comput. Sci. 231, 17–32.

Mohri, M., Pereira, F., Riley, M., 2002. Weighted finite-state transducers in speech recognition. Comput. Speech Language 16 (1), 69–88.

Padmanabhan, M., Saon, G., Basu, S., Huang, J., Zweig, G., 1999. Recent improvements on a voicemail transcription task. In: Proc. EUROSPEECH'99. Budapest.

Palmer, D.D., 2001. Modeling uncertainty for information extraction from speech data, Ph.D. Thesis, University of Washington.

Palmer, D.D., Ostendorf, M., Burger, J.D., 1999. Robust information extraction from spoken language data. In: Proc. EUROSPEECH'99. Budapest.

Rahim, M., Riccardi, G., Saul, L., Wright, J.H., Buntschuh, B., Gorin, A. L., 2001. Robust numeric recognition in spoken language dialogue. In: Speech Communication, vol. 34. pp. 195–212.

Roark, B., 2002. Markov parsing: lattice rescoring with a statistical parser. In: Proc. 40th ACL Meeting.Philadelphia.

Ron, D., Singer, Y., Tishby, N., 1998. On the learnability and usage of acyclic probabilistic finite automata. J. Comput. Syst. Sci. 56 (2), 133-152. URL: citeseer.nj.nec.com/ron951earnability.html.

Stolcke, A., Omohundro, S., 1994. Inducing probabilistic grammars by bayesian model merging. In: Int. Conf. Grammatical Inference. URL: citeseer.nj.nec.com/stolcke94inducing.html.

Tur, G., Wright, J.H., Gorin, A.L., Riccardi, G., Hakkani-Tur, D., 2002. Improving spoken language understanding using word confusion networks. In: Proc. ICSLP'02. Denver, Colorado.