On the Development of Telephone Applications: Some Practical Issues and Evaluation

A. Facco\textsuperscript{(1)}, D. Falavigna\textsuperscript{(1)}, R. Gretter\textsuperscript{(1)}, M. Vigano\textsuperscript{(2)}

\textsuperscript{(1)} ITC-Irst, Istituto per la Ricerca Scientifica e Tecnologica, Trento (Italy)
\textsuperscript{(2)} Reitek Spa, Milano (Italy)
(facco, falavi, gretter)@itc.it, m.vigano@reitek.com

Abstract

This paper describes the major problems encountered during the development of two automatic services delivered through the telephone platform reported in [1]. Service specification, system design and development, and service evaluation and tuning will be described. Given an initial set of user trials, a significant part of the work was devoted to the refinement of the language models (actually a combination of both regular grammars and tri-grams).

System performance has been evaluated with different sets of acoustic models, including both context independent and context dependent phone models.

1. Introduction

Speech community, both research and industry, is well aware of the difficulty to develop reliable automatic telephone services based on speech recognition. Even for simple Interactive Voice Response (IVR) services, particular care must be devoted in application designing, specifically for defining both the dialogue strategy and voice prompts. As an example, during the development of the Italy Direct service [2], we observed that user answers to the voice prompt “say yes or no” were strictly “yes” or “no” solely in 44% of the cases. This percentage rose to 67% when the voice prompt was changed to “say clearly yes or no”.

Although some statistical methods have recently been proposed for either automatically learning the dialogue strategy [3], or for modeling hierarchical structures in the language [4] in order to decode semantic classes, there is no evidence about their effectiveness for developing real telephone applications.

Our experience suggests that, from a purely practical point of view, the rapid and successful development of applications still relies on the usage of modular architectures, based on components (e.g. basic speech recognition grammars), easily reusable across different domains. The dialogue engine and related development tools, described in [5], has proven to be effective for delivering telephone applications as the ones reported below. Also the mixed language model, proposed in [6], allows improving grammar effectiveness and portability.

The possibility to discard sentences not covered by speech recognition grammars, e.g. by means of either rejection networks or confidence measures, allows improving system usability, provided that a reasonable balance between false rejections and false acceptances can be achieved. In the same way, the capability to recognize speech during a voice message play (“barge-in”) adds flexibility to services, as well as the ability of detecting and, possibly, correcting errors [7].

In the following, we will describe two telephone services that use some of the above mentioned features. Then, we will report the related performance and discuss the possible benefits brought by each feature.

Both services are delivered through the telephone platform described in section 2. A significant number of user interactions, including both speech signals and recognizer outputs, have been collected during the usage of the systems. These corpora form the test/training sets employed in the experiments.

Acoustic and language models, together with procedures and algorithms utilized for their training and refinement will be described in section 3. In one of the two services (i.e. the automatic payment of the road tax) both context independent and context dependent acoustic models have been compared. In this service, furthermore, an unsupervised training procedure, similar to the one reported in [8], has been evaluated.

The second service, a voice portal for accessing financial information, is mainly based on word spotting. We will discuss the effort needed to both develop and tune the related speech recognition grammars.

Experiments and results will be given in section 4.

2. System architecture and service design

The system manages speech interactions during communications by means of both a multi-client multi-server architecture.

The telephone infrastructure consists of: the telephone front-end, one or more speech servers, and the enterprise back-end. The telephone front-end manages user interface. Speech servers perform Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) functions. Enterprise back-end hosts database and legacy applications.

Speech resources can be concurrently accessed by several client applications located on one or more platforms. Resource allocation is controlled by a centralized manager, implementing load balancing algorithms, and providing redundancy mechanisms for achieving high availability and scalability.

Other main features are barge-in capability and VoiceXML support.

2.1. Service specifications

The two telephone services examined in this work are: 1) a speech IVR application, delivered by Automobile Club Italia (ACI), for the automatic payment of the road tax; 2) a banking voice portal, delivered by UNICREDIT bank, for accessing some of the existing customer services.

In the ACI service most of the information (e.g. menu choices, confirmations, dates, credit card numbers, etc.) has to be provided in DTMF mode, with the exception of the car plate, which is alphanumeric. At first, users are invited to pronounce their car plates freely and, in case of recognition errors, using city names. After three consecutive errors, users are transferred to a human agent. Some examples of car plate sentences are: “A B one two three C D”, “A as Ancona B as Bologna one hundred twenty three C as Catania D”; etc. Furthermore, speech recognition grammars can take into account some additional expressions, such as: “hi”, “good morning”, “the plate number is”, “my plate is”, etc. It is worth noting that this service allows to automatically manage about 50% of the incoming calls.
UNICREDIT voice portal allows users to: obtaining information on bank’s products and services, retrieving investment funds quotations, knowing the balance of their current accounts, asking for a fax containing the list of recent transactions, checking the status of their orders, managing their access credentials and stop their credit cards. For other services, or in case of recognition errors, callers can choose either to use DTMF or to be transferred to a human agent.

For this service, examples of possible user utterances are: “I want a fax with the investment funds quotation”, “I need the balance of my account”, “I want to listen to the list of my last 10 transactions”, etc.

At present, the system manages 180 telephone lines with 50 ASR ports and 70 TTS ports. It addresses three vertical markets: retail, private and small business. The service tailored for small business customers has been available since December 2003. In the next few months, services for private and retail customers will also start.

2.2. Dialog design

In both of the services described above, the design and deployment of the Voice User Interface (VUI) has required not only traditional systems analysis and software development skills but also the specialized skills of: linguists, speech scientists, human factors specialists and business process analysts.

Relying on these additional skills, we approached the task of producing both of the two mentioned speech applications from the point of view of the traditional software development life cycle. After an accurate requirement analysis, we defined interactive dialogs keeping in account user logic, rather than application design logic. In the implementation phase, we created user-friendly messages in order to pronounce suitable prompts with regards to the call flow.

The application test has been split into two parts: user testing and pilot assessment. For user testing, we gave a small selected group of people (about 20) the task of using the service: the behaviour and feedback gathered from these people provided key information for validating the accuracy of the dialog. For pilot assessment, we tested the application on a broader basis in a real-life situation. This phase, still in progress in the UNICREDIT service, is essential to guarantee a good level of user satisfaction. Statistical analysis of the collected audio traces is vital in the UNICREDIT service, is essential to guarantee a good level of user satisfaction. Statistical analysis of the collected audio traces is essential to guarantee a good level of user satisfaction.

3. Acoustic and language models

3.1. Acoustic models

Context Independent (CI) acoustic models correspond to a rich set of phone units (140 in total), which includes specific models for alphadigits, confirmations, background noise and hesitations.

Context Dependent (CD) models, namely triphones, have been built with a procedure similar to the one described in [10]. A decision tree based clustering algorithm of HMM output distributions allows reaching an optimal compromise, in a maximum likelihood sense, between specificity and trainability of the resulting CD models (see [10] for the details). We point out that CD models allow modeling both word internal and cross word context dependencies.

For the ACI task it was possible to collect a considerable amount of user interactions (about 20 hours). Given this size of the material we investigated the possibility to adapt, in a completely unsupervised way (or with a small manual effort), the general purpose CI models to the specific task. We developed an automatic selection algorithm in order to find out an “optimal” (in some sense) subset of task specific data to be added to the baseline database for training models. The chosen optimality criterion was that to minimize the Word Error Rate (WER); alternatively, the Word Error Rate (WER) could be used.

The core of the algorithm lies in the usage of rejection grammars (alternatively, confidence measures could be used). A rejection step is carried out on a set of task specific data by putting a rejection network in parallel with the recognition grammar. This allows discarding a number of speech segments exhibiting a probability ratio value exceeding a predefined threshold, where rej and rec are rejection and recognition grammars, respectively. Note that the rate of the discarded segments can be controlled by varying the value of language model probability assigned to the rejection network itself (the higher this value the higher the number of discarded segments). Then, a retraining phase is led on the non discarded material and the WER is evaluated on a separated development set. This process can be iterated as explained below.

Let us define the following quantities: $M_i$ the acoustic model set used at step $i$,  $M_0$ the baseline acoustic model set, $T_i$ the task specific retraining set at step $i$, $B$ the baseline (task independent) training set, $T$ the initial task specific retraining set, $D$ a task specific development set and $P_i(M_i,D)$ the word error probability given $M_i$ and $D$. The unsupervised retraining algorithm proceeds according to the following steps.

1. $i=0$; set $M_i = M_0$; set $T_i = T$;
2. evaluate $P_i(M_i,D)$;
3. transcribe $T_i$ using $M_i$;
4. retrain $M_{i+1}$ using $B$ and $T_i$;
5. evaluate $P_i(M_{i+1},D)$;
6. decrease the rejection language model probability;
7. discard from $T_i$ all segments giving a probability ratio greater than one and generate $T_{i+1}$;
8. if $P_i(M_{i+1},D) > P_i(M_i,D)$ then return with $M_{i+1}$ and $T_{i+1}$; else $i = i+1$; goto 4.

3.2. Language models

The language model makes use of recurrent transition networks [9]. With this approach the recognizer output contains both the words...
and the semantic information of the input utterance. As an example, a possible output for an utterance of the ACI task can be:

**GREETINGS** (Hallo) **GREETING** (my car plate is) **NUMBER** (123) **NUMBER**

The example above can be interpreted as a parse tree, where semantic labels are shown in bold. Terminal symbols (i.e., words inside semantic labels) can be generated by either regular grammars or stochastic grammars (e.g., n-grams). From example above, note that the two models (i.e., regular and stochastic grammars) can be combined in a completely general way. However, in the experiments reported below, we have used trigrams only for modeling part of sentences of the financial information service (UNICREDIT) while, for the ACI service, we have only used regular grammars.

### 4. Experiments and results

The baseline acoustic training database comes from various sources and includes about 37 hours of speech. It contains phonetically rich sentences, digit and confirmation utterances, alpha-digits and the telephonic part of a large broadcast news database [11]. About 16 hours of the whole set are the telephonic filtered version of 16 kHz speech, acquired in quiet environment.

#### 4.1. ACI task

For the ACI task it was possible to initially collect a significant set of telephone calls (about 4 hours of speech signals). This material was automatically transcribed, manually checked and then used to refine speech recognition grammars. Successively, a further set of task specific data, TS (18.2 hours of speech in total), was recorded. Then, we applied to TS the unsupervised selection algorithm reported in section 3.

Both test (400 speech files, 46 minutes) and development (146 speech files, 13 minutes) sets have been automatically transcribed and manually checked. Then, 4 iterations of the unsupervised retraining procedure were run, obtaining subsets TS$_1$ (16 h), TS$_2$ (13.7 h), TS$_3$ (11.2 h) and TS$_4$ (9.7 h). Results are given in table 1. In the table, BL represents the supervised baseline training database.

<table>
<thead>
<tr>
<th>training</th>
<th>dev: WER / SER</th>
<th>test: WER / SER</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL</td>
<td>10.41% / 44.52%</td>
<td>12.80% / 44.25%</td>
</tr>
<tr>
<td>BL U TS</td>
<td>9.26% / 39.72%</td>
<td>10.48% / 41.25%</td>
</tr>
<tr>
<td>BL U TS$_1$</td>
<td>8.68% / 39.04%</td>
<td>9.99% / 39.25%</td>
</tr>
<tr>
<td>BL U TS$_2$</td>
<td>8.20% / 36.98%</td>
<td>9.85% / 39.00%</td>
</tr>
<tr>
<td>BL U TS$_3$</td>
<td>7.91% / 34.93%</td>
<td>9.89% / 38.50%</td>
</tr>
<tr>
<td>BL U TS$_4$</td>
<td>8.68% / 37.67%</td>
<td>10.17% / 39.00%</td>
</tr>
</tbody>
</table>

Note in the table, the benefit brought by the proposed unsupervised training procedure (from 12.8% WER to 9.85% WER, about 20% relative WER reduction).

To further assess the effectiveness of this procedure we decided to apply it for selecting, from TS, an optimal subset of files to be employed in a supervised training step. The criterion adopted for deriving this subset was that to remove, from TS, as many sentences (or part of sentences) as possible that are either correctly recognized or that contain non alpha-digit words. To do this, we analyzed a small subset of files (50 in total) extracted from TS, TS$_1$, TS$_2$ and TS$_3$. We noticed that files belonging to TS both give a large percentage of recognition errors and contain a large number of non alpha-digit words. Files belonging to both TS$_1$ and TS$_2$ provide a large number of recognition errors and contain a small number of non alpha-digits. Finally, files belonging to TS$_3$ only give few recognition errors. This suggests to select, for supervised training, the subset TSR=(TS$_1$ \ TS$_3$), where operator \ denotes difference between sets. After having trained on BLUTS$_R$ (TS$_R$ is formed by 4.8 hours of automatically transcribed and manually checked files) we obtained the results given in table 2.

<table>
<thead>
<tr>
<th>training</th>
<th>dev: WER / SER</th>
<th>test: WER / SER</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL U TSR</td>
<td>7.43% / 34.24%</td>
<td>8.13% / 26.75%</td>
</tr>
</tbody>
</table>

Comparing performance in table 2 with the baseline performance in table 1, one can note again the effectiveness of the proposed procedure, even for selecting the material to be used for supervised training.

In experiments with context dependent units, we trained HMMs in a completely supervised way, using the set BLUTSUTS$$_R$; TS$_R$ is an additional task specific training set consisting of 8.5 hours of speech. Results obtained with both context independent and context dependent models are given in table 3.

<table>
<thead>
<tr>
<th>models</th>
<th>WER</th>
<th>SER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI</td>
<td>8.68%</td>
<td>30.60%</td>
</tr>
<tr>
<td>CD</td>
<td>5.98%</td>
<td>24.73%</td>
</tr>
</tbody>
</table>

The improvement achieved with CD models is similar to those obtained on other different tasks reported in the literature [10]; however, it is worth noticing the additional memory and computation requirements with respect to CI models. Table 4 gives the number of models (#mod), number of Gaussian distributions (#distr), the corresponding memory occupation (in Mbytes) and computation time (in seconds) for decoding the whole test set of the ACI task.

<table>
<thead>
<tr>
<th>models</th>
<th>#mod</th>
<th>#distr</th>
<th>MByte</th>
<th>sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI</td>
<td>140</td>
<td>3724</td>
<td>2.5</td>
<td>480</td>
</tr>
<tr>
<td>CD</td>
<td>9450</td>
<td>58177</td>
<td>40</td>
<td>3500</td>
</tr>
</tbody>
</table>

Looking at table 4, it follows that the major impact in using CD models lies in the increase of computation time. For a given dimension of the telephone platform, basically depending on the number of telephone lines, this requires augmenting the number of computation resources by a factor of almost 7.

#### 4.2. UNICREDIT task

In this service users, instead of being presented a long menu, are invited to freely ask for the desired information. Then, either word or phrase spotting are performed on input utterances in order to only extract the semantically relevant words.

Also in this case, an initial set of sentences (887 in total, about 39.4 minutes of speech signals) was collected from trials carried out by about 20 bank employers. Then, several word spotting solutions, based on an optimal usage of rejection grammars, were tested.

All of the test sentences were manually transcribed at both orthographic and semantic level. For instance, the sentence “I want to listen to the list of last transactions” contains the two semantic tokens: [VOICE (to listen to) VOICE] and [TRANSACTION (transactions) TRANSACTION].
The remaining words (i.e., “I want” and “the list of last”) have to be rejected. The whole test set contains 2486 words (2.8 words per sentence, on average) and 769 semantic units.

An initial grammar was defined consisting in a loop of all the phrases to spot. We noticed that 265 of the test sentences (29.9%) were covered by the grammar itself; the remaining 622 sentences (70.1%) were out of the grammar coverage and, therefore, need to be totally, or partially, rejected. Consequently, we defined several rejection grammars, to be used in opposition with the phrases to spot, with the purpose of balancing the number of false alarms and recognition errors on the test set. Furthermore, since the test database is quite small, a leave-one-out technique was adopted, consisting in splitting the test sentences into two subsets. One subset (development set, formed by 443 sentences) was used for estimating grammar parameters (e.g., rejection probabilities), and the other (test set, formed by 444 sentences) for measuring performance. In a successive step, the roles of development and test sets are exchanged and the overall performance is evaluated combining the results achieved on the two leave-one-out test sets.

Table 5 shows performance, expressed in terms of semantic accuracy, obtained on the UNICREDIT task. In the table, SSR represents string recognition rate, URR represents unit recognition rate, #Err is the total number of recognition errors, #D+#I+#S gives the number of deletion, insertion and substitution errors.

<table>
<thead>
<tr>
<th>Rejection Grammar</th>
<th>SSR</th>
<th>URR</th>
<th>#Err</th>
<th>#D+#I+#S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone loop</td>
<td>72.60%</td>
<td>61.90%</td>
<td>293</td>
<td>25 + 237 + 31</td>
</tr>
<tr>
<td>Tuned phone loop</td>
<td>79.84%</td>
<td>74.25%</td>
<td>198</td>
<td>65 + 103 + 30</td>
</tr>
<tr>
<td>Garbage grammar</td>
<td>82.53%</td>
<td>77.89%</td>
<td>170</td>
<td>43 + 108 + 19</td>
</tr>
<tr>
<td>Unigrams</td>
<td>83.77%</td>
<td>79.46%</td>
<td>158</td>
<td>52 + 78 + 28</td>
</tr>
<tr>
<td>Trigrams</td>
<td>87.94%</td>
<td>82.05%</td>
<td>138</td>
<td>35 + 77 + 26</td>
</tr>
</tbody>
</table>

The baseline rejection grammar (first raw in table 5) simply consists in a phone loop. To reduce the number of insertion errors (237), provided by this grammar, we estimated an optimal weight (“penalty”) to be added to the overall log-likelihood, evaluated during the Viterbi search, before entering the rejection network. Doing this, URR raised from 61.9% to 74.3% (second raw in table 5).

To further improve system performance we built a garbage grammar containing all of the words in the development sets not belonging to valid phrases. This grammar was again put in opposition to the other two grammars, enabling the recognition of any combination of valid phrases, rejected phones and garbage words. This allowed increasing URR till 77.9% (third raw in table 5). A further URR improvement was achieved (79.5%, fourth raw in table 5) by adding unigram probabilities to the garbage grammar.

In the last experiment, we processed the sentences in the development sets in order to estimate a trigram language model. As an example, the sentence “I want to listen to the list of last transactions” is transformed into “I want VOICE the list of last TRANSACTION”, where VOICE and TRANSACTION are semantic classes. These training data allow building stochastic grammars that includes both words and semantic classes. Best URR performance (82.05%) was obtained with this approach.

5. Conclusions and discussion

We have described some of the problems encountered during both the design and development of two telephone applications.

We have demonstrated the effectiveness of an automatic selection procedure to be used for either unsupervised or lightly supervised retraining. In spite of this fact, most of the effort was spent for usability test, which proved to be both time consuming and expensive. Human factors, statistical analysis, evaluation of dialog success rates and tuning on field data, strongly affect the overall costs of services.

Although barge-in adds flexibility to services, its usage increases system complexity, and makes dialogs less comfortable every time the system has to recover from wrongly detected barge-in events.

The usage of CD units was also shown to be effective in one of the two reported services, at the cost of an increase of about one order of magnitude of the computation resources.

6. Acknowledgements

We thank Automobile Club Italia and Unicredit Bank for having rendered available the speech material used in the experiments. Part of this work has been funded by the EU project Homey.

7. References