Improved Large Vocabulary Speech Recognition Using Lexical Rules

K. Georgila[1], A. Tsopanoglou[2], N. Fakotakis[1], G. Kokkinakis[1]

(1) Wire Communications Lab.
University of Patras, 265 00 Patras, Greece
Tel.: +30 61 991722 Fax.: +30 61 991855
{rgeorgil, fakotaki, gkokkin}@wcl.ee.upatras.gr

(2) Knowledge S.A., LogicDIS Group
N.E.O. Patron-Athinon 37, 264 41, Patras, Greece
Tel. +30 61 452820 Fax. +30 61 453819
atsopano@knowledge.gr

SUMMARY

Some applications of speech recognition, such as automatic directory information services, require very large vocabularies, e.g. surnames, first names, city names etc. In this paper, we focus on the task of recognizing surnames in an Interactive telephone-based Directory Assistance Services system[1], which supersedes other large vocabulary applications in terms of complexity and vocabulary size. We aim at refining the N-best hypotheses' list provided by a speech recognizer by applying lexical rules. Thus a small value of N produces multiple solutions and therefore it becomes sufficient to retain high accuracy and at the same time achieve real-time response. Experimental results have proved the efficiency of the approach. By applying the rules, the accuracy of the speech recognizer has risen from 71.08% to 87.83% for N=5, a value that also ensures that we have fast performance in both cases.

KEYWORDS : Large vocabulary, name recognition, lexical rules, N-best hypotheses.

INTRODUCTION

In several applications such as automatic directory information services, a speech recognizer is expected to be able to handle very large vocabularies. In this paper, we focus on the task of recognizing surnames of a public directory since it supersedes other large vocabulary applications in terms of complexity and vocabulary size. However, the proposed method can be applied to every kind of large vocabulary.

In order to stress the importance of the problem and show its connection to real-world applications, we provide some examples of existing automatic directory information systems and we examine how each one addresses the problem. The Philips Automatic Directory Information System [3], handles the directory of the city of Aachen, Germany, and vicinity. Its 131,001 listings include 38,608 distinct surnames. In the first turn, the user is asked to spell out the desired surname. At that time, the search space consists of the full database, but the recognizer is limited to spelling and the number of possible surnames extracted is usually significantly less than 100. In the subsequent dialogue turns, the user is asked to utter the surname, the first name, and finally the street name, one after the other. The search space is reduced with every dialogue turn.

In the British Telecom Automatic Directory Assistance Service [4] the dialogue model is somehow different. The caller is asked to give the town and the road name first. Then the system prompts the user to utter the desired surname and its spelling. During the development of their system, British Telecom experimented with all sorts of dependencies and reached the conclusion that if recognitions stay independent of each other and the N-best lists are intersected with the database, confidence increases while accuracy drops. On the other hand, if successive recognitions are constrained by previous ones, accuracy gets higher and confidence decreases.

If there was no dialogue turn for spelling in the aforementioned systems and the caller was prompted directly to utter the surname first, the value of N in the N-best hypotheses' list of the speech recognizer would have to be high. This would ensure that the correct surname (the one uttered by the user) is included. There are many acoustically similar surnames, and if N is small it is very likely that the correct surname does not appear in the list because the N positions of the list are all occupied by surnames acoustically similar to the correct surname. However, a very high value of N would slow down the system's response. In this paper, we present a method to avoid using spelling and at the same time retain high performance with a low value of N.

In our application, the European Project IDAS [1], the recognizer must distinguish between 257,198 distinct surnames that correspond to 5,303,441 entries in the directory of the Greek Organization of Telecommunications. By restricting the search space to the most frequent 88,000 ones that correspond to about 123,000 distinct pronunciations, 93.57% of the directory's listings is covered. Each dialogue turn is independent of the previous ones. Therefore the search space is not reduced with every dialogue turn. First the user is asked to utter the town. Then the caller is

[1]The EU Project LE-48315, IDAS.

prompted to fully utter the surname and give its first letter. In Greek, spelling is not usual (splitting the word in syllables is preferred), and thus it is not used in the dialogue system. The speech recognizer searches among all the distinct pronunciations of surnames, which start with the letter given by the user, producing the N-best hypotheses as output. Then the phonetic transcription of the surnames is transformed to the graphemic one. Note that a single phonetic transcription could lead to multiple graphemic ones. The above transformation is done automatically since both the phonetic and graphemic transcriptions of a surname are stored in the lexical database. Otherwise a phoneme-to-grapheme converter would be used. After the phonetic transcription of the surname has been transformed to the corresponding graphemic one, lexical rules are applied, which define classes of letters and letter combinations, the members of which can be falsely recognized in a specific context. That is a letter or letter combination of a class could be mistaken for another letter or letter combination of the same class in a specific context defined by the rule. The solutions created by applying the lexical rules are surnames acoustically similar to the N-best hypotheses produced by the speech recognizer. The rules are language dependent and they are carefully selected so that they cover the most probable interchanges between letters or letter combinations, but without leading to too many solutions.

A similar approach has been applied to letter recognition in [2]. The spoken letters processed by a free letter recognizer generate a list of N-best hypotheses. Each hypothesis is converted to a sequence of letter classes that are used to search a tree. That is, acoustically similar letters have been grouped to form a letter class and each letter has been replaced by the name of the class where it belongs. Starting at the root of the tree, the class sequence specifies a path to a leaf that contains names similar to the input letter hypotheses. The concatenation of names across all N-best leaves provides a short list of candidates that can be searched in more detail in the rescoring stage using either letter alignment or an acoustic search using a tightly constrained grammar.

The paper is organized as follows: The structure of the rules is described in the following section. Then the algorithm, which processes the above rules is explained and results of performed experiments are provided. Finally, some conclusions are given in the final section.

RULES' STRUCTURE

The structure of the rules is as follows:

$L_1, L_2 \ldots L_k, S, R_1, R_2 \ldots R_n$

where L_i i=1, … k is the left context of the rule, S is the class, which includes letters or letter combinations that could be interchanged, and R_p p=1…n is the right context of the rule. The values of k and n could vary according to the language and the way the designer of the rules has decided to form them. Each L_i or R_p is a class of letters or letter combinations that could substitute one another as context of the central part of the rule S. In our experiments we have selected k=1 and n=3, which means that we look only one class of letters or letter combinations backwards and 3 forward. Nevertheless, the processing algorithm is parametric and could work for any values of k and n. Thus a rule could have the following form for k=3 and n=3:

-, -, -, ΓΚ Κ, W, NULL, NULL (Rule 1a)

where NULL stands for any step not considered by the rule and the dash for an empty string. The above structure will be transformed to

-, ΓΚ Κ, W, NULL, NULL (Rule 1b)

for k=1 and n=3. Note that it is very important to place k steps before the central part of the rule so that the algorithm understands which the central part is. However, putting in the right context fewer steps than n, will not affect the rule. Therefore rules 1a and 1b are equivalent to rules 1c and 1d respectively.

-, -, -, ΓΚ Κ, W (Rule 1c)

-, ΓΚ Κ, W (Rule 1d)

Rules 1a-1d state that ΓΚ can be interchanged with Κ, when no letter precedes them and when they are followed by any letter or letter combination contained in cluster W. We are not interested in what follows after W and that is what the 2 NULL symbols denote in rules 1a and 1b. W is defined as

W=(Α,Β,Γ,Δ,Ε,Ζ,Η,Θ,Ι,Κ,Λ,Μ,Ν,Ξ,Ο,Π,Ρ,Σ,Τ,Υ,Φ,Χ, Ψ,Ω,-)

that is cluster W includes all the letters of the Greek alphabet plus the dash, which is used when we are at the beginning of a word (left context) or at the end (right context). Thus the previous rule could be applied if for example Α follows ΓΚ or Κ and the word starts with ΓΚ or Κ. The dash shows that before ΓΚ or Κ, we have an empty string, which means that we are at the beginning of the word. The use of clusters prevents us from having too many rules e.g.

-, ΓΚ Κ, Α

-, ΓΚ Κ, Ε

etc.

In the same way we have the following rule:

W, ΤΣΙ ΤΣ, W (Rule 2a)

That is ΤΣΙ and ΤΣ are interchanged in all cases regardless of what precedes or follows. If we used k=2 then the previous rule could be transformed to

NULL, W, ΤΣΙ ΤΣ, W (Rule 2b)

or

W, ΤΣ, Ι -, W (Rule 2c)

The above example shows that the values of k and n depend on both the language and the decisions made by the designer of the rules regarding their structure. In the following we consider k=1 and n=3. According to the rule

W, ΑΣ Α, -, -, - (Rule 3a)

ΑΣ and Α are interchanged when they are last in a word. Rule 3a is equivalent to Rule 3b:

W, ΑΣ Α, - (Rule 3b)

Another option in the rules' structure is depicted in the following example:

W-, Ρ(V1)Γ ΡΓ, W (Rule 4)

where

W-=
(Α,Β,Γ,Δ,Ε,Ζ,Η,Θ,Ι,Κ,Λ,Μ,Ν,Ξ,Ο,Π,Ρ,Σ,Τ,Υ,Φ,Χ,Ψ,Ω)

and

V1=(Α,Ε,Ι,Η,Υ,ΕΙ,ΟΙ,ΥΙ,ΑΙ,Ο,Ω,ΟΥ)

Rule 4 says that ΡΓ can be interchanged with ΡΑΓ, ΡΕΓ, ΡΙΓ, ΡΗΓ, ΡΥΓ, ΡΕΙΓ, ΡΟΙΓ, ΡΥΙΓ, ΡΑΙΓ, ΡΟΓ, ΡΩΓ, ΡΟΥΓ in a specific context. This context is class W-, which means that any letter could precede a letter or letter combination included in the central part of the rule, apart from the empty string, that is the rule is not applied when we are at the beginning of a word. The right context is class W, which means that any letter can follow. Cluster V1 contains all the vowels and double vowels of the Greek language. However, this leads to invalid combinations such as ΡΕΙΓ, ΡΟΙΓ, ΡΥΙΓ, ΡΑΙΓ. Nevertheless sometimes we prefer to have broad clusters so that they are not specific for one rule but not too broad to avoid multiple invalid solutions, which will lead to increasing the system's response time. In the same way cluster W in rules 1a-1d leads to invalid combinations e.g. ΓΚΠ but we use it to avoid having too many different clusters and to prevent the designer of the rules from omitting some rare cases of letter combinations. That is, if the designer tried to make clusters that would include only the appropriate (not redundant) letters or letter combinations for a specific context, it is very likely that s/he would fail to consider all the cases for this particular context.

Other examples of rules' structures are:

V, Κ Π Τ ΓΚ ΝΤ ΜΠ, V, Σ, - (Rule 6)

V, Κ Π Τ ΓΚ ΝΤ ΜΠ, V, -, - (Rule 7a)

V, Κ Π Τ ΓΚ ΝΤ ΜΠ, V, - (Rule 7b)

where V=(Α,Ε,Η,Ι,Ο,Υ,Ω,ΟΥ). Rules 7a and 7b are equivalent.

RULES' PROCESSING

The algorithm that processes the rules in order to produce acoustically similar words, works as follows: each one of the solutions (input strings to our algorithm) given by the speech recognizer is processed. Each input string is traversed from the first letter until the last one. When a letter or a letter combination is the same as the central symbol in the rule, then the rule is applied and new strings are created. The pointer in the input string moves forward as many positions as the ones denoted by the central part of the rule. The procedure does not stop when the condition for the application of the first appropriate rule is met. It continues until all possible rules are applied. An example is described in the following. Suppose that the recognizer has given the output

ΚΑΤΣΙΑΟΥΝΟΣ

which is the input string to our algorithm and we have rules

W, ΤΣΙ, ΤΣ, W (Rule 1)
W, ΤΣ, ΤΖ, W (Rule 2)
W, (V)ΓΟΥ (V)ΟΥ, W (Rule 3)
-, ΓΚ Κ, W (Rule 4)

where

W=(Α,Β,Γ,Δ,Ε,Ζ,Η,Θ,Ι,Κ,Λ,Μ,Ν,Ξ,Ο,Π,Ρ,Σ,Τ,Υ,Φ,Χ, Ψ,Ω,-)

and

V=(Α,Ε,Ι)

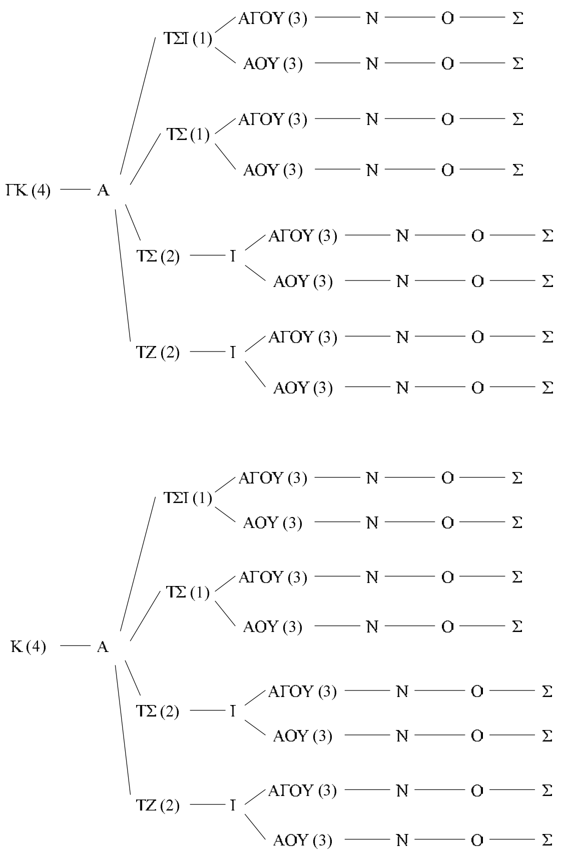

Figure 1: The application of rules for the input string ΚΑΤΣΙΑΟΥΝΟΣ (without look-up in the lexicon).

The first rule says that ΤΣΙ can be interchanged with ΤΣ in any context. The second rule denotes that ΤΣ and ΤΖ can replace one another again in any context. The third rule states that ΑΓΟΥ, ΕΓΟΥ, ΙΓΟΥ can be interchanged with ΑΟΥ, ΕΟΥ and ΙΟΥ respectively in any context. Finally, according to the fourth rule, ΓΚ can be replaced by Κ or vice versa when ΓΚ or Κ are first in a word. The procedure of the rules' application is depicted in Figure 1. The numbers in parentheses show the rule that is applied each time. The input string ΚΑΤΣΙΑΟΥΝΟΣ is traversed from left to right. The first letter is Κ. The algorithm searches for a rule where Κ is one of the central symbols. The 3 first rules cannot be applied but the fourth one can. Thus we have two solutions so far:

ΓΚ	(A1)
Κ	(A2)

We go back to the input string. The pointer moves to Α. Again the algorithm will search for an appropriate rule. However, no rule can be applied so the pointer moves to Τ and the resulting strings so far are:

ΓΚΑ	(B1)
ΚΑ	(B2)

Now that we are in Τ, the first rule is applied and we get

ΓΚΑΤΣΙ	(C1)
ΓΚΑΤΣ	(C2)
ΚΑΤΣΙ	(C3)
ΚΑΤΣ	(C4)

The pointer moves to Α. We go on to find if another rule is applicable. The second rule is, so we get 4 additional solutions:

ΓΚΑΤΣ	(C5)
ΓΚΑΤΖ	(C6)
ΚΑΤΣ	(C7)
ΚΑΤΖ	(C8)

The pointer moves to Ι for solutions C5-C8, but it was placed at Α for C1-C4. Consequently we have to store different pointers according to the positions in the input string, where different rules are applied. Now the algorithm processes each one of the 8 solutions we have so far. Rule 3 applies to solutions C1-C8, so we get

ΓΚΑΤΣΙΑΓΟΥ	(D1)
ΓΚΑΤΣΙΑΟΥ	(D2)
ΓΚΑΤΣΑΓΟΥ	(D3)
ΓΚΑΤΣΑΟΥ	(D4)
ΚΑΤΣΙΑΓΟΥ	(D5)
ΚΑΤΣΙΑΟΥ	(D6)
ΚΑΤΣΑΓΟΥ	(D7)
ΚΑΤΣΑΟΥ	(D8)

The pointer moves to Ν. For solutions C5-C8 the pointer is at Ι and no rule is applied so we get:

ΓΚΑΤΣΙ	(D9)
ΓΚΑΤΖΙ	(D10)
ΚΑΤΣΙ	(D11)
ΚΑΤΖΙ	(D12)

The pointer is positioned at Α. The procedure continues until we reach the end of the input string. The final solutions are:

ΓΚΑΤΣΙΑΓΟΥΝΟΣ	(E1)
ΓΚΑΤΣΙΑΟΥΝΟΣ	(E2)
ΓΚΑΤΣΑΓΟΥΝΟΣ	(E3)
ΓΚΑΤΣΑΟΥΝΟΣ	(E4)
ΚΑΤΣΙΑΓΟΥΝΟΣ	(E5)
ΚΑΤΣΙΑΟΥΝΟΣ	(E6)
ΚΑΤΣΑΓΟΥΝΟΣ	(E7)
ΚΑΤΣΑΟΥΝΟΣ	(E8)

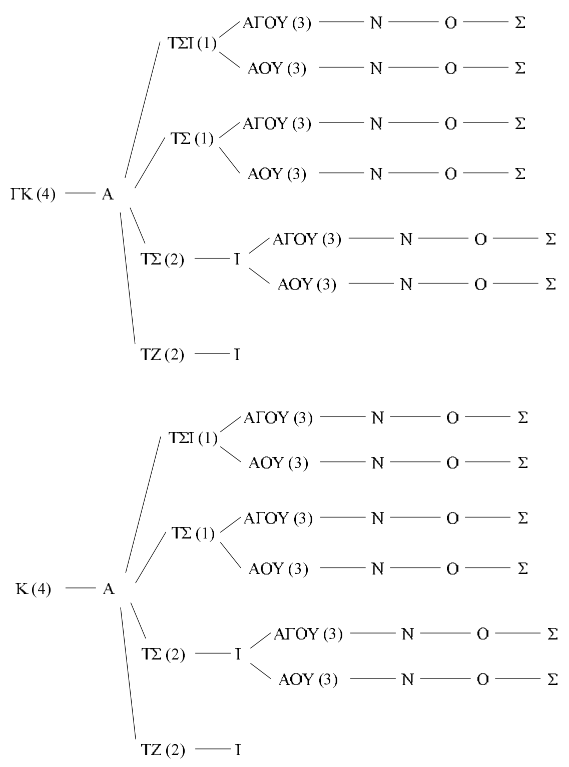

Figure 2: The application of rules for the input string ΚΑΤΣΙΑΟΥΝΟΣ (with look-up in the lexicon).

ΓΚΑΤΣΙΑΓΟΥΝΟΣ	(E9)
ΓΚΑΤΣΙΑΟΥΝΟΣ	(E10)
ΓΚΑΤΖΙΑΓΟΥΝΟΣ	(E11)
ΓΚΑΤΖΙΑΟΥΝΟΣ	(E12)
ΚΑΤΣΙΑΓΟΥΝΟΣ	(E13)
ΚΑΤΣΙΑΟΥΝΟΣ	(E14)
ΚΑΤΖΙΑΓΟΥΝΟΣ	(E15)
ΚΑΤΖΙΑΟΥΝΟΣ	(E16)

Note that some solutions are identical, e.g. E1 and E9, E2 and E10, E5 and E13, E6 and E14. This does not constitute a problem since the redundant strings will be discarded before the final search in the database. That is, the system will first look up the solutions in the lexicon of distinct surnames and it will discard the invalid ones. Finally it will search for the remaining solutions in the telephone directory. The reason the algorithm does not search for identical strings each time new solutions are produced by the application of rules, is in order to be as fast as possible. Some other things have to be taken into consideration as well. If Rule 1 had the following form:

W, ΤΣ ΤΣΙ, W (Rule 1a)

then the algorithm would first find the substring ΤΣ, then it would replace it with ΤΣΙ and move the pointer to Ι. Therefore instead of ΓΚΑΤΣΙΑ, ΓΚΑΤΣΑ, ΚΑΤΣΙΑ and ΚΑΤΣΑ we would get ΓΚΑΤΣΙΙΑ, ΓΚΑΤΣΙΑ, ΚΑΤΣΙΙΑ and ΚΑΤΣΙΑ. To avoid this problem, either we put in the centre of the rule the substrings according to their length, or we modify the algorithm so that it takes into account the length of the symbols.

Some rules may produce words that do not exist and are not included in the database. It is desirable and saves much processing time to stop extending a substring if we realize that it would not lead to valid solutions. Thus the system looks up a solution in the lexicon of distinct surnames if its length has exceeded the threshold of 4 letters. If no word that begins with this substring exists, the solution is abandoned. The reason we start looking up the solutions in the lexicon only when their length is greater than 4 is that normally it takes more than 4 letters to decide whether a surname is valid or not. If we started looking up the solutions in the lexicon from the first letter, this would increase the processing time with no benefit. In the previous example if the system does not find any words starting with e.g. ΓΚΑΤΖ or ΚΑΤΖ it will abandon all the solutions produced by applying rules to ΓΚΑΤΖ or ΚΑΤΖ. In Figure 2 we can see how the rules are applied and some paths are abandoned when they lead to invalid solutions. Again as in Figure 1, the numbers in parentheses indicate the applied rule.

Currently the rules are extracted manually. However, a technique based on Hidden Markov Models (HMM) for automatically extracting the rules from the list of the most frequent surnames is under development, which will improve further the efficiency of the proposed method. Statistical processing of the list of most frequent surnames has also produced weights for each rule. Suppose that we have rules 1 and 2 (see the Rules' Processing section). We find N1 surnames that would be similar if we interchanged ΤΣΙ and ΤΣ in any context W, and N2 surnames that would be equivalent if we replaced ΤΣ with ΤΖ and vice versa again in any context W. If N1>N2 then Rule 1 has a greater weight than Rule 2. The weights of the rules that have been used to produce a solution are combined with the confidence of the source hypothesis (the one were rules were applied) provided by the speech recognizer, to give the confidence of the new solution. Thus in the end, after we discard the invalid solutions by looking them up in the lexicon of distinct surnames, we have all the valid surnames with their confidence levels and we are ready to search in the telephone directory.

EXPERIMENTAL RESULTS

Tests were carried out with 110 people. The 76 males called the system 381 times and the 34 females 123 times. The recognition accuracy for the surnames before the application of lexical rules was 70,85%. In that case the speech recognizer produced only the best hypothesis (N=1). There was also an additional stage, where the system asked the user to confirm whether the recognized surname was correct or not. A speech synthesizer was used to speak out the recognized surname. However, the low quality of speech synthesis caused problems. In total we had 105 missed calls. 51 of them (10.12% of the total calls) arose from the fact that the recognizer recognized correctly the uttered surname but when the synthesizer pronounced this name and asked for confirmation, the user did not understand that the name uttered was the correct one and gave a negative confirmation. On the other hand 10 calls (1,98% of the total calls) were missed because the recognizer produced an invalid surname. This surname was uttered by the synthesizer, the user thought that the correct name was pronounced and gave a positive confirmation.

The above results led us to use different approaches in order to improve the speech recognition accuracy. To carry out the new tests 37 people (23 male and 14 female) uttered 10 different surnames each, that is we had 370 surnames to be recognized in total. We experimented with different values of N and both with and without lexical rules. The results are depicted in Table 1. The first line shows the absolute values and the second the % percentage.

	Male	**Female**	**Total**
Without Lexical Rules			
N=1	159 69.13%	98 70.00%	257 69.46%
N=5	163 70.87%	100 71.43%	263 71.08%
N=10	168 73.04%	102 72.85%	270 72.97%
N=20	179 77.82%	108 77.14%	287 77.56%
N=30	191 83.04%	116 82.85%	307 82.97%
With Lexical Rules			
N=1	195 84.78%	119 85.00%	314 84.86%
N=5	202 87.82%	123 87.85%	325 87.83%
N=10	207 90.00%	127 90.71%	334 90.27%

Table 1: Experimental results.

If we do not use lexical rules, the best results are given when the recognizer produces the 30-best hypotheses. However, in this case the response time is quite increased, which necessitates a lower value of N. We have not kept record of the response time in all these tests. Nevertheless, it was obvious that the system stopped being real-time with N greater than 5 because the computational cost became too high. When we applied lexical rules, we realized that N=1 was enough to produce better results than N=30 (with no lexical rules) and with no significant computational cost. This was due to the fact that the cost of processing the signal in order to produce multiple outputs is much higher than the time, which is required in order to take an output and apply the lexical rules. Moreover, the application of lexical rules leads to a lot more than 30 solutions, which have the advantage to be based on language dependent data and not only on the acoustic signal and thus the probability of including the correct surname is higher. The results are even better when we have N=10 and use lexical rules. But in this case, as for N=10 without rules, the response time is not very good. In conclusion N=5 with lexical rules is the solution that combines good recognition accuracy and real-time response.

CONCLUSIONS

In this paper we described a technique applied to large vocabulary speech recognition. This method aims at refining the N-best hypotheses' list provided by a speech recognizer by applying lexical rules. The performed experiments showed that our approach, that is the application of rules, results in better recognition accuracy compared to the cases when no rules are applied, for the same value of N or even when N is smaller in the first case. That is the accuracy for N=1 when rules are applied is better than the accuracy for N=30 without rules. Moreover, the computational cost is much smaller, which leads to real-time response without sacrificing accuracy.

Currently the rules are formed manually, so our future work focuses on developing an algorithm for their automatic extraction. In this way, we expect that we will cover cases not captured by the human designer and at the same time we will completely automate the process. Further experiments will be carried out concerning the optimization of the trade-off between recognition accuracy and response time.

BIBLIOGRAPHY

1. European Project LE-48315, IDAS.
2. Mitchell, C.D. and Setlur, A.R. *Improved spelling recognition using a tree-based fast lexical match.* In Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'99), Phoenix, AZ.
3. Seide, F. and Kellner, A. *Towards an automated directory information system.* In Proc. of the 5th European Conference on Speech and Communication Technology (Eurospeech'97), Vol. 3, pp. 1327-1330, Rodos, Greece, Sept. 22-25.
4. Whittaker, S.J. and Attwater, D.J. *Advanced speech applications – the integration of speech technology into complex services*. ESCA workshop on Spoken Dialogue Systems – Theory and Application, pp. 113-116, Visgo, June 1995.